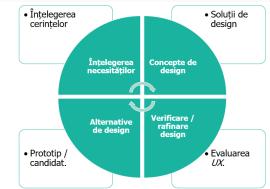




Curs 2 - UX/UI design

1 play • 49 players

A private kahoot



Questions (6)

1 - Quiz

Despre care dintre activitățile fundamentale ale UX design a fost acest curs?

- | | | |
|-------------------------------------|------------------------|--|
| <input checked="" type="checkbox"/> | înțelegerea cerințelor | |
| <input type="checkbox"/> | soluții de design | |
| <input type="checkbox"/> | prototipare | |
| <input type="checkbox"/> | evaluarea UX | |



2 - Quiz

Ce se realizează preferabil înainte de vizita la companie/client?

20 sec

- | | | |
|-------------------------------------|--------------------------------|--|
| <input checked="" type="checkbox"/> | Documentare despre domeniu. | |
| <input type="checkbox"/> | Interviuri și observare. | |
| <input type="checkbox"/> | Stabilirea parametrii vizitei. | |
| <input type="checkbox"/> | Alcătuirea echipei. | |

3 - Puzzle

Ordonați cele patru etape parcuse pentru înțelegerea cerințelor

60 sec

-  Colectarea datelor
-  Analiza datelor
-  Modelarea datelor
-  Extragerea cerințelor

4 - Quiz

Care etapă este (îndeosebi) deductiv-analitică?

20 sec

-  Modelarea datelor ✗
-  Analiza datelor ✗
-  Extragerea cerințelor ✓
-  Colectarea datelor ✗

5 - Quiz

Un protocol de lucru este legat îndeosebi de...

20 sec

-  domeniul de activitate ✗
-  practicile de lucru ✓
-  activitățile unei persoane ✗
-  altceva ✗

6 - True or false

În cadrul discuțiilor pentru înțelegerea cerințelor designer-ul evită, pe cât posibil, să impună propriile opinii.

20 sec



True



False

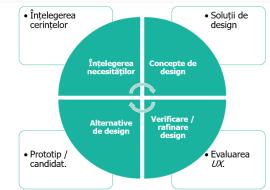




Curs 3 - UX/UI design

1 play • 27 players

A private kahoot

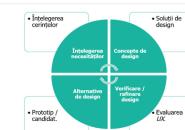


Questions (7)

1 - Quiz

Despre care dintre activitățile fundamentale ale UX design a fost acest curs?

- Înțelegerea cerințelor
- soluții de design
- prototipare
- evaluarea UX



✓

✗

✗

✗

2 - True or false

Imaginea de mai jos este reprezentativă pentru WAAD (Work Activity Affinity Diagram)

- True
- False

✓

✗

3 - Puzzle

Ordonați cele patru etape parcuse pentru înțelegerea cerințelor

60 sec



- Colectarea datelor
- Analiza datelor
- Modelarea datelor
- Extragerea cerințelor

4 - Quiz

Alegeți etapa care a fost discutată astăzi.

20 sec



- Modelarea datelor ✗
- Analiza datelor ✓
- Extragerea cerințelor ✗
- Colectarea datelor ✗

5 - Quiz

Work activity notes trebuie să fie...

20 sec



- lungi ✗
- dependente de alte notițe ✗
- concise ✓
- modulare ✓

6 - Quiz

Am căutat pe Internet păreri despre activitățile de la acest club, nu am reușit să aflu mare lucru este...

20 sec



requirement



user story



7 - Quiz

Indicați care dintre afirmații corespunde unui requirement

20 sec



Este greu să obțin informații despre sporturile practicate în acest club



Mă interesează îndeosebi evenimentele organizate la sfârșit de săptămână



Aș dori să pot sorta după categorii (sporturi individuale, de echipă, etc)



Am citit o serie de recenzii foarte interesante despre acest club





Curs 1 - UX/UI Design

1 play • 19 players

A private kahoot



Questions (8)

1 - Quiz

Acest curs este (în primul rând) despre...

20 sec

- Programare orientată pe obiecte ✗
- Programare web ✗
- Experiența de utilizare ✓
- Teoria grafurilor ✗

2 - Quiz

La word cloud-ul creat, cele mai multe opțiuni au fost pentru

20 sec

- a ✗
- v ✗
- c ✗
- c ✓

3 - Quiz

Eficiență și ușurință de utilizare sunt legate de...

20 sec

- Utilizabilitate ✓

- Utilitate ✗

- Impact emoțional ✗

- Meaningfulness ✗

4 - Quiz

În ce an a avut loc întâlnirea de referință *Demarcating User eXperience Seminar?*

20 sec

- 1970 ✗

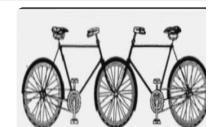
- 1999 ✗

- 2010 ✓

- 2020 ✗

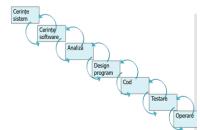
5 - True or false

Imaginea de mai jos era legată de *meaningfulness* (persistență în timp)



- True ✗

- False ✓



6 - Quiz

În imagine este prezentat un model...

- liniar ✗
- în cascadă ✓
- circular ✗
- de tip *wheel* ✗

7 - Quiz

Bifați activitățile principale ale fluxului de lucru din *UX* design

20 sec

- înțelegerea cerințelor ✓
- soluții de design ✓
- realizarea de prototipuri ✓
- evaluarea *UX* ✓

8 - Quiz

În sensul prezentării din slide-uri, luarea de notițe este...

20 sec

- o activitate ✗
- o sub-activitate ✗
- o metodă ✗
- o tehnică ✓

To appear in Behaviour and Information Technology. Do not copy or cite without permission.

Manuscript cover page

**Cognitive, Physical, Sensory, and Functional Affordances
in Interaction Design**

H. Rex Hartson

Department of Computer Science – 0106

Virginia Tech

Blacksburg, VA 24061

Phone: 540/231-4857

Fax: 540/231-6075

email: hartson@vt.edu

Cognitive, Physical, Sensory, and Functional Affordances in Interaction Design

H. Rex Hartson

Department of Computer Science – 0106

Virginia Tech

Blacksburg, VA 24061

540/231-4857, hartson @vt.edu

...they may indeed look, but not perceive, and may indeed listen, but not understand

Mark 4.12 (NRSV)

Abstract

In reaction to Norman's [1999] essay on misuse of the term affordance in human-computer interaction literature, this article is a concept paper affirming the importance of this powerful concept, reinforcing Norman's distinctions of terminology, and expanding on the usefulness of the concepts in terms of their application to interaction design and evaluation. We define and use four complementary types of affordance in the context of interaction design and evaluation: cognitive affordance, physical affordance, sensory affordance, and functional affordance. The terms cognitive affordance (Norman's perceived affordance) and physical affordance (Norman's real affordance) refer to parallel and equally important usability concepts for interaction design, to which sensory affordance plays a supporting role. We argue that the concept of physical affordance carries a mandatory component of utility or purposeful action (functional affordance). Finally, we provide guidelines to help designers think about how these four kinds of affordance work together naturally in contextualized HCI design or evaluation.

1. Introduction

Reacting to his urge to speak up while lurking among CHI-Web discussants over-using and misusing the term affordance, Don Norman was compelled to explain the concept of affordance in his essay [1999], ‘Affordance, conventions, and design’. We¹ agree with most of what Norman said, but feel there is more to be said about the concept of affordance, especially to the end of making it a useful and applicable concept for usability designers and practitioners. Since Norman encouraged it in his opening paragraph: ‘Hope it doesn’t stop the discussion again’ [Norman, 1999], we decided to add to the discussion, affirming the importance of this powerful concept, reinforcing Norman’s distinctions of terminology, and adding some of our own ideas about applying affordance to interaction design and evaluation.

1.1. The importance of semantics and terminology

This is a concept paper, not a methodology paper or a report of an empirical study. The epistemological cycle in the science of human-computer interaction (HCI), as in most disciplines, alternates empirical observation with theory formulation to explain and predict the observed. Norman’s stages of action model [1986] is a practical example of HCI theory, in that it explains and predicts what users do while interacting with systems (from refrigerators to computers) to accomplish goals in a work domain. It is our intention here to develop more fully some key concepts as a contribution to that kind of HCI theory.

In essence this paper is about semantics and terminology to express semantics. HCI is a relatively young field and the terminology we require for discussing, analyzing, and applying

¹ Although this is a single-author paper, most of it is written in first person plural to acknowledge much help from many HCI colleagues at Virginia Tech.

our concepts with a common understanding is incomplete. The terms we use for concepts are not inherently important, but the semantics behind the terminology commands our attention. In response to, ‘It’s just semantics,’ we heartily agree with Allen and Buie [2002, p. 21] who proclaim: ‘Let us say it outright: There is no such thing as *just* semantics. . . . In communication, nothing is more important than semantics.’ Allen and Buie [2002, p. 18] are dead on: ‘This isn’t just nit-picking—a rich and evocative word like *intuitive* is wasted as long as it sits in a fog of uncertain associations.’ This statement was never more true than it is for the term affordance, as Norman’s essay [1999] attests. Shared meanings and representations (through common language) are an absolute must in science, art, and everything in-between.

1.2. Gibson on affordance

Norman begins by referring to Gibson’s earlier definitions of afford and affordance [1977; 1979], as well as to discussions he and Gibson have had about these concepts. Setting a paraphrase of Gibson [1979, p. 127] within an HCI design context, affordance as an attribute of an interaction design feature is what that feature offers the user, what it provides or furnishes. Here Gibson is talking about physical properties, what Norman calls real affordances. Gibson gives an example of how a horizontal, flat, and rigid surface affords support for an animal. In his ecological view, affordance is reckoned with respect to the user, in this case the animal, who is part of the affordance relationship. Thus, as Norman [1999] points out, Gibson sees an affordance as a physical relationship between an actor (e.g., user) and physical artefacts in the world reflecting possible actions on those artefacts. Such an affordance does not have to be visible, known, or even desirable.

1.3. Norman on affordance

In his article, Norman [1999] takes issue with a common and growing misuse (or perhaps uninformed use) of the term affordance. In simple terms, much of the difficulty stems from confusion between what Norman calls real affordance and perceived affordance. To Norman [1999], the unqualified term affordance refers to real affordance, which is about physical characteristics of a device or interface that allow its operation, as described by Gibson in the previous section. However, in many HCI and usability discussions the term is also used without qualification to refer to what Norman calls perceived affordance, which is about characteristics in the appearance of a device that give clues for its proper operation. Since the two concepts are very different, perhaps orthogonal, Norman admonishes his readers not to misuse the terms and, in particular, not to use the term affordance alone to refer to his concept of perceived affordance and, perhaps, not to use these terms at all without understanding the difference.

1.4. Seeking a balance for interaction designers

In these admonishments [1999], Norman focuses mainly on real affordance. We believe that what Norman calls perceived affordance has an equally important role, perhaps even a starring role, in interaction design. We know that Norman believes this, too. In his book Design of Everyday Things [Norman, 1990], sometimes called the DOET book – formerly Psychology of Everyday Things [Norman, 1988], known as the POET book – Norman describes his struggles with refrigerators, British water taps, and other physical devices and says much about perceived affordances in the context of problems that users of these devices have in determining how to operate them. Norman feels that DOET might have played a part in the confusion of terms because, as he says [1999], ‘I was really talking about perceived

affordances, which are not at all the same as real ones'. However, in the course of emphasizing the difference in his more recent article, we feel that the importance of perceived affordances became somewhat lost, leaving researchers and practitioners in a quandary about how we can legitimately refer to this important usability concept. In hopes of a remedy we offer a perspective on the concept of affordance that has been working for us. We would like to strike a balance and we think Norman would approve.

1.5. Objectives

We think it is healthy when an article like Norman's leads to a follow-up discussion, especially about a topic essential to interaction design. In that spirit, this is not a critique or rebuttal. Rather, Norman has called for understanding of these concepts, and has highlighted the problem of inadequate terminology. We wish to respond to that call by suggesting terminology for four kinds of affordance without violating Norman's or Gibson's basic precepts but, in fact, amplifying and extending them in a useful way. Like Norman, we would like to see these concepts understood and properly distinguished in their use by researchers and practitioners alike. In the process, we would also like to give Norman credit for a broader contribution in his stages-of-action model [Norman, 1986] than perhaps he may have given himself.

We have named the different kinds of affordances for the role they play in supporting users during interaction, reflecting user processes and the kinds of actions users make in task performance. Norman's perceived affordance becomes cognitive affordance, helping users with their cognitive actions. Norman's real affordance becomes physical affordance, helping users with their physical actions. We add a third kind of affordance that also plays an

important role in interaction design and evaluation, sensory affordance, helping users with their sensory actions. A fourth kind, functional affordance, ties usage to usefulness. We offer guidelines for considering these kinds of affordance together in a design context.

2. Related work

2.1. *Calibrating terminology*

Since Norman brought the term affordance into common usage in the HCI domain with his book Design of Everyday Things [Norman, 1990], the term has appeared many times in the literature. For example, an interesting recent treatment by Thimbleby shows how key aspects can be formalised as mathematical symmetry [2002].

In this section, we show the relationships among others' use of the terminology and ours. In so doing, we give a preview of our definitions and usage, along with a rationale for our particular choices.

Beyond Gibson and Norman, McGrenere & Ho [2000] and Gaver [1991] have influenced our thinking about affordances. McGrenere & Ho [2000] give credit to Gibson for originating the concept of affordance in psychology and to Norman for introducing this important concept into human-computer interaction. McGrenere & Ho also target current misuse and confusion of terms, noting the need to clarify the concepts for effective communication among researchers and practitioners and make a connection to usability design. Gaver [1991] sees affordances in design as a way of focusing on strengths and weaknesses of technologies with respect to the possibilities they offer to people who use them. Gaver also summarizes his view of the Gibson and Norman contributions. He extends the concepts by showing how complex actions can be described in terms of groups of affordances, sequential

in time and/or nested in space, showing how affordances can be revealed over time, with successive user actions, for example, in the multiple actions of a hierarchical drop-down menu. That McGrenere and Ho [2000] also needed to calibrate their terminology against Gaver's further demonstrates the difficulty of discussing these concepts without access to a richer, more consistent vocabulary. Table 1 shows how various authors use the terminology, compared to usage in this paper.

Table 1. Comparison of affordance terminology

	Hartson	Physical affordance	Cognitive affordance	Sensory affordance
Gibson	Affordance	Perceptual information about an affordance	Implied	
	Real affordance	Perceived affordance	Implied	
	Affordance	Perceptual information about an affordance	Indirectly included in perceptibility of an affordance	
Gaver	Affordance, also perceptible affordance	Perceptual information about an affordance, also apparent affordance	Indirectly included in perceptibility of an affordance	

In most of the related literature, design of cognitive affordance (whatever it is called in a given paper) is acknowledged to be about design for the cognitive part of usability, ease-of-use in the form of learnability for new and intermittent users (who need the most help in knowing how to do something). But the concept gets confused because a cognitive affordance is variously called a perceived affordance, an apparent affordance, or perceptual information about an affordance.

What McGrenere & Ho and Gaver simply call an affordance and what Norman calls a real affordance is, by and large, what we call a physical affordance, offered by artefacts that can be acted upon or physically manipulated for a particular purpose. All authors who write

about affordances give their own definitions of the concept, but almost no one, including Norman [1986] (who, to be fair, intended to focus on the cognitive side) and McGrenere & Ho [2000] (e.g., in their Section 6.2), mentions design of physical affordances. Design of physical affordances is about design for the physical action part of usability, ease-of-use in the form of high performance and productivity for experienced and power users as well as to help disabled users achieve maximum efficiency in physical actions. McGrenere & Ho come close to recognizing this role of physical affordance in design in the discussion about their Figure 4, which relates cognitive affordance and physical affordance to design improvement.

Most other authors, including those in Table 1, include sensory affordance only implicitly and/or lumped in with cognitive affordance rather than featuring it as an separate explicit concept. Thus, when these authors talk about perceiving affordances, including Gaver's and McGrenere & Ho's phrase 'perceptibility of an affordance', they are referring (in our terms) to a combination of sensing (e.g., seeing) and understanding physical affordances through sensory affordances and cognitive affordances. Gaver refers to this same mix of affordances when he says, 'People perceive the environment directly in terms of its potential for action'. As we explain in the next section, our use of the term 'sense' has a markedly narrower orientation on sensory inputs such as seeing and hearing.

2.2. Level setting

Why maintain separate terms and concepts when they are to be integrated in design, anyway? The answer is simply that the differences among these concepts requires that each type of affordance must be identified for what it is and considered on its own terms in analysis and design. Each type of affordance plays a different role, uses different mechanisms,

corresponds to different kinds of user actions, exhibits different characteristics, has different requirements for design, and implies different things in evaluation and diagnosis.

In this section we articulate a rationale for boundaries in the particular use of psychological terminology in the context of affordances, guided by a motivation to clearly bring out issues of HCI design and analysis. The concepts of sensing, perception, and cognition all have a large scope in their broadest interpretation, too broad for isolating the HCI design factors of affordances. In the general context of psychology, these concepts are more intertwined than orthogonal. To avoid this intertwining we use, for example, the term ‘sensing’ instead of ‘perception’ in most places, because perception usually embraces significant cognition [Hochberg, 1964]. Our motivation for attempting a degree of arbitrary compartmentalization, via reasonable operational definitions that work on a practical level for design, is that the HCI design issues we wish to associate with these levels of user actions are mostly orthogonal.

While overlapping and borderline cases are interesting to psychologists, HCI designers want to avoid marginal design and ensure that designs work for wide-ranging user characteristics. An abstraction that separates the types of user actions (e.g., sensing from cognition) removes the overlap. As an illustration, consider text legibility, which at a low level is about identifying shapes in displayed text as letters in the alphabet, but not about the meanings of these letters as grouped into words and sentences. Text legibility is an area where user perception, sensing, and cognition can overlap. To make out text that is just barely or almost barely discernable, users can augment or mediate sensing with cognition, using inference and the context of words in a message to fill in the blanks. Context can make some candidate

letters more likely than other. Users can recognize words in their own language more easily than words in another language or in nonsense letter combinations.

In contrast, HCI design in this context requires solutions resolved on the side of pure sensing. Simply put, a label in a user interface that cannot be fully discerned by the relevant user population, without reliance on cognitive augmentation, is a failed HCI design. Thus, we wish to define sensing at a level of abstraction that eliminates these cases of borderline user performance so that HCI designers can achieve legibility, for example, beyond question for the target user community. We desire an understanding of affordance that will guide the HCI designer to attack a text legibility problem by adjusting the font size, for example, not by adjusting the wording to make it easier to deduce text displayed in a tiny font.

In our abstraction, a user's sensory experience can include gestalt aspects of object appearance and perceptual organisation [Arnheim, 1954; Koffka, 1935], such as figure/ground relationships, and might sometimes include some judgment and lexical and syntactic interpretation in the broadest spatial or auditory sense (e.g., what is this thing I am seeing?), but does not get into semantic interpretation (e.g., what does it mean?). In the context of signal processing and communications theory, this kind of sensing would be about whether messages are received correctly, but not about whether they are understood.

A discussion of HCI design without the kind of abstraction we propose can degenerate to hair splitting about levels of human information processing that distract from the practical design issues, further putting off practitioners who may already believe that concepts like affordance are just fodder for academic exercises.

3. Our proposal

To pursue the objectives of Section 1.5, specifically in the context of interaction design and evaluation for computer-based systems, we propose (the essence of the value-added in this article):

1. to clarify and define the terms cognitive affordance and physical affordance to refer to parallel and equally important usability concepts for interaction design,
2. that the concept of physical affordance carries a mandatory component of utility or purpose, which we call functional affordance, and that statements about physical affordance must include a reference to that purpose,
3. to add the concept of sensory affordance, supporting cognitive affordance and physical affordance in design, and
4. that cognitive, physical, sensory, and functional affordance be connected and considered together in any HCI design or evaluation context.

3.1. Cognitive and physical affordance – an alliance in design

The relevant part of what my dictionary says about ‘to afford’ is that it means to yield, to give, or to furnish. In design, an affordance gives or provides something that helps a user do something. For example, the study window in my house affords me a fine view of the forest; the window helps me see that nice view. Norman’s stages-of-action model [1986] describes the typical course of interaction between a human user and a computer or any kind of machine. During interaction, a user performs cognitive, physical, and sensory actions and requires affordances to help with each. In our work on the User Action Framework [Andre, Hartson, Belz, & McCreary, 2001; Andre, Belz, McCreary, & Hartson, 2000; Hartson, Andre, Williges, & van Rens, 1999], based on Norman’s model, we have also found a need

for all four kinds of affordance in the context of interaction design and usability. It is in that context that we offer these definitions.

A cognitive affordance is a design feature that helps, aids, supports, facilitates, or enables thinking and/or knowing about something. As a simple example, clear and precise words in a button label could be a cognitive affordance enabling users to understand the meaning of the button in terms of the functionality behind the button and the consequences of clicking on it. A physical affordance is a design feature that helps, aids, supports, facilitates, or enables physically doing something. Adequate size and easy-to-access location could be physical affordance features of an interface button design enabling users to click easily on the button. Since physical affordance occurs with physical objects, I am treating active interface objects on the screen, for example, as real physical objects, since they can be on the receiving end of real physical actions by users. As many in the literature have pointed out, it is clear that a button on a screen cannot be pressed. Restricting the discussion to clicking on buttons easily dispatches this difficulty.

Norman [1999, p. 41] says that symbols and constraints are not affordances and that wording in the label on a button, for example, is symbolic communication. We agree, but under our definition, communication is exactly what makes good wording effective as a cognitive affordance: something to help the user in knowing (e.g., knowing what to click on). We see symbols, constraints, and conventions as essential underlying mechanisms that make cognitive affordances work, as Norman says, as ‘powerful tools for the designer’. As Norman further says, the only way we know for sure if users share designers’ perceptions of these symbols and conventions is by usability data. Thus, and we think this is a point that

Norman particularly had in mind in his article, if a designer claims to have ‘added an affordance’ to the interaction design, that in itself says nothing about usability.

In the DOET [Norman, 1990] tradition, we illustrate with a simple and ubiquitous non-computer device, a device for opening doors. The hardware store carries both round doorknobs and lever type door handles. The visual design of both kinds conveys a cognitive affordance helping users think or know about usage through the implied message their appearance gives to users: ‘This is what you use to open the door’. The doorknob and lever handle each suggests, in its own way, the grasping and rotating required for operation. Again, we agree with Norman in noting that the message implied is based on convention and there is nothing intrinsic in the appearance of a doorknob that necessarily conveys this information. On another planet, it could seem mysterious and confusing, but for us a doorknob is an excellent cognitive affordance because almost all users do share the same easily recognized cultural convention.

Door operation devices also provide physical affordance, to help users do the opening and closing – some better than others. For example, many users prefer the lever type to a round knob because the lever is easier to use with slippery hands or by an elbow when the hands are full. The push bar on double doors is another example of a physical affordance helpful to door users with full hands.

Sometimes the physical affordance to help a user open a door is provided by the door itself; people can open some swinging doors by just pushing on the door. In such cases designers often help users by installing, for example, a brass plate to show that one should push and where to push. Even though this plate might help avoid handprints on the door, it is a

cognitive affordance and not a real physical affordance, because it adds nothing to the door itself to help the user in the physical part of the pushing action. Sometimes the word ‘Push’ is engraved in the plate to augment the clarity of meaning of the plate as a cognitive affordance.

Similarly, sometimes the user of a swinging door must open it by pulling. The door itself does not usually offer sufficient physical affordance for the pulling action, so a pull handle is added. A pull handle offers both cognitive and physical affordance, providing a physical means for pulling as well as a visual indication that pulling is required.

Norman discusses many such everyday devices in his DOET book [1990] and makes it clear that, when he speaks of knowing how to operate a device, he is referring to cognitive (perceived, in his terminology) affordance, characterizing a view of cognitive affordance that we share [Norman, 1999, p. 39]: ‘When you first see something you have never seen before, how do you know what to do? The answer, I decided, was that the required information was in the world: the appearance of the device could provide the critical clues required for its proper operation’. However, when Norman later says that affordances play a relatively minor role in the world of screen-based systems [1999, p.39], he clearly is talking about physical affordances (and the statement is true only if one is not concerned with design factors for physical actions, such as those involving Fitts’ law [MacKenzie, 1992], physical disabilities, or the physical characteristics of interaction devices). And we think Norman would agree that cognitive affordances play an enormously important role in interaction design; cognitive affordances are one of the most significant user-centred design features in present-day interactive systems, screen-based or otherwise. They are the key to answering Norman’s question: ‘How do you know what to do?’ And, yes, the design of cognitive

affordances can depend greatly on cultural conventions as a common base for communicating the meaning of visual cues from designer to user.

Continuing in the DOET tradition of non-computer examples, we have known many different kinds of wine bottle openers, possessing a range of effectiveness. Although most people understand how to use the kinds of openers shown in Figure 1, their design could offer better physical affordance, to help the user in doing the physical task for which they were intended. Because of somewhat crude mechanical operation, they often manage to crumble the cork, leaving bits unappetizingly bobbing in the newly liberated libation.



Figure 1. Ordinary cork pullers with acceptable cognitive affordance

In contrast a colleague, Roger Ehrich, recently gave me the marvelously efficient and reliably effective cork puller shown in Figure 2.



Figure 2. A cork puller with good physical affordance but non-obvious cognitive affordance

The problem with this device, though, is that its proper use was initially anything but obvious to me. For the sake of science, we have been increasing the frequency of informal user-based tests and find that an average of more than nine out of ten wine-drinking guests who have not seen this design before cannot determine how to use it in a reasonably short time. This device offers excellent physical affordance to help in doing the task, making it a good design for an experienced user such as a wine steward. However, it does not offer good cognitive affordance for helping intermittent and first-time users know or learn how to use it.

The secret to operation lies in shifting between modes in a classic case of moded design: there are two states, and in each state user actions and inputs have meanings and outcomes that are different from those of the other state (see Chapter 11 of Thimbleby [1990]). The thick piece of metal connecting the T-handle to the threaded shaft, at the left of Figure 2, is what makes this opener different from most others. By swiveling, it functions as a kind of ‘gear shift’ that changes the way the threads are engaged, lending the modality to the design.

Figure 3 shows the T-handle moved to the top of the threaded shaft and the shifting mechanism in the centre has locked the T-handle to a fixed position on the threaded shaft. When the T-handle is rotated in this cork engagement mode, the threaded shaft turns and the corkscrew at the bottom dives deftly into the cork.



Figure 3. T-handle locked to threaded shaft in the cork engagement mode

Then the shifting mechanism is swiveled, unlocking the T-handle from the shaft, putting the device in lifting mode. When the T-handle is now turned (in the same direction as before), the shaft does not turn but the T-handle moves along the threads on the shaft, lifting the cork, as in Figure 4.



Figure 4. T-handle moving on the shaft threads in lifting mode

3.2. Functional affordance – design for purposeful action

The second part of our proposal is to bring Gibson's ecological view into contextualized HCI design by including a purpose in the definition of each physical affordance. Putting the user and purpose of the affordance into the picture harmonizes nicely with our interaction- and user-oriented view in which an affordance helps or aids the user in doing something.

As Norman points out [1999, p. 40], his own definition of (physical) affordance means that all interface designs afford clicking anywhere on the screen, whether a button is there or not, except where the pointer is constrained from being in certain parts of the screen (a hypothetical condition that Norman introduced to make his point). But that kind of clicking is without reference to a purpose and without the requirement that any useful reaction by the system will come of it. But, of course, we need more than that in a task-oriented context of interaction design, where user actions are goal-oriented and purposeful. A user doesn't click on the screen just because it's possible. A user clicks to accomplish a goal, to achieve a purpose (e.g. clicking on a user interface object, or artefact, to select it for manipulation or clicking on a button labeled 'Sort' to invoke a sorting operation). So, if designers or users say this button affords clicking, we would understand this to mean the button is sensitive to clicking in the sense that the system will usefully respond to clicking. But even this interpretation does not go far enough to meet our proposed requirement to associate purpose with physical affordance. It is more useful to be specific about the purposeful response and say that the button affords clicking to initiate, for example, the Sort function. A blank space on the screen next to the button, or another button elsewhere on the screen, does not provide that same kind of physical affordance. Adding the purpose for a physical affordance adds sense and a goal orientation to a design discussion.

The study window in my house affords me a fine view of the forest, but I have to participate by looking through the window to accrue the benefit of seeing that view. Gibson implicitly included reference to purposeful enablement: a horizontal, flat, rigid surface affords an animal to stand, walk, or run. Gibson is indeed talking here about purposeful activity, as he

is in the discussion [1979], for example, about how objects (artefacts) afford manipulation (e.g. a pole that can be used by a chimpanzee as a rake to reach a banana).

In Norman's DOET world of non-computer devices, a purpose for a physical affordance is always implied. The doorknob is a cognitive and physical affordance for operating the door. The physical affordance offered by a doorknob does not mean merely that the doorknob can be grasped and turned. It means that the doorknob can be grasped and turned in order to operate (e.g. invoke the function or mechanism of opening) the door; the user is enabled to operate the door. In turn, the door itself is a functional affordance that, when invoked, allows passage. In this interaction design view, a physical affordance gives access to functionality, the *purpose* of the physical affordance used to access it.

McGrenere & Ho [2000] also refer to the concept of application functionality usefulness, something they call 'affordances in software'. In an external view it is easy to see a system function as an affordance because it helps the user do something in the work domain. This again demonstrates the need for a richer vocabulary, and conceptual framework, to take the discussion of affordances beyond user interfaces to the larger context of overall system design. We use the term *functional affordance* to denote this kind of higher-level user enablement in the work domain.

As McGrenere & Ho [2000] point out, requiring purposeful action as a component of physical affordance nicely substantiates the dual concepts of usability and usefulness [Landauer, 1995]. Usefulness stems from the utility of functional outcomes of user actions. In contrast, usability stems from the effectiveness of cognitive affordances for understanding

how to use physical affordances, from the physical ease of using the physical affordances, and from the sensing of these via sensory affordances.

In sum, the addition of purpose to the description of a physical affordance is an obvious extension, but it should be made explicit, to avoid the ambiguities Norman has described. This extension to the concept of physical affordance might possibly go beyond what either Gibson or Norman had in mind, but we think it makes sense and is not difficult to justify in the domain of design.

3.3. *Sensory affordance – a supporting role*

The third part of our proposal is to include the concept of sensory affordance. A sensory affordance is a design feature that helps, aids, supports, facilitates, or enables the user in sensing (e.g., seeing, hearing, feeling) something. Sensory affordance includes design features or devices associated with visual, auditory, haptic/tactile, or other sensations. Cognitive affordance and physical affordance are stars of interaction design but sensory affordance plays a critical supporting role. In short, sensory affordance can be thought of as an attribute of cognitive affordance or physical affordance; users must be able to sense cognitive affordances and physical affordances in order for them to aid the user's cognitive and physical actions. Sensing cognitive affordances is essential for their understanding, and sensing physical affordances is essential for acting upon them. Sensory affordance issues of user interface artefacts include noticeability, discernability, legibility (in the case of text), and audibility (in the case of sound). In the concept of sensory affordance, as we explained in Section 2.2, we have deliberately included only the physical act of sensing and not any of the cognitive aspects that are often associated with the term perception.

3.4. Contextualized design as nexus of affordance roles

The fourth part of our proposal is to connect all four kinds of affordance in a design context.

To put Gibson's ecological view in HCI terms, affordances have a relational ontology: their existence as an affordance is relative to the environment of users and usage. In HCI, the user's environment is the work context plus the interaction design. To accomplish work goals, the user must sense, understand, and use affordances within an interaction design.

Gaver [1991] says that affordances are a powerful approach for thinking about technology because the effectiveness of an affordance depends on the attributes of both the artefact and the user. The concept of affordance is an instrument for focusing on links in design among the user, the actions, and the artefacts. The user's path from sensing to cognition to action shows how each affordance role is involved in both learning about (ease of learning) and using (ease of use) artefacts. The idea is to include both user and artefact attributes in affordance designs as part of the complementarity that Gaver describes, between actor and acted-upon environment. Gaver's ecological perspective offers a succinct approach to artefact design through an immediate connection between cognitive and physical affordances.

In Gestalt psychology [Koffka, 1935], well before Gibson or Norman, we see the connection of cognitive affordance to physical affordance and its purpose [Gibson, 1979, p. 138]. The meaning or value or use of a thing can be seen and understood through that object (at least if an effective design or cultural convention supports it), just as one can see its size or colour. Similarly, we design human-computer interaction for the user to understand the operation and purpose of a physical affordance through sensing (via sensory affordances) and understanding associated cognitive affordances.

Norman [1999, p. 41] says, ‘Affordances (*meaning physical affordances*) specify the range of possible activities, but affordances are of little use if they aren’t visible to the users’, meaning ‘visible’ in both the sensory (detectable or observable) and cognitive sense (understandable). Physical affordance is associated with the ‘operability’ characteristics of user interface artefacts. Cognitive affordance is associated with semantics or meaning of user interface artefacts. Sensory affordance is associated with the ‘sense-ability’ characteristics of user interface artefacts, especially of physical affordances and cognitive affordances. In the domain of human-computer interaction, as in the domain of everyday physical devices, design is what connects physical affordances to the cognitive affordances that ‘advertise’ them and explain how to use, when to use, and whether to use each physical affordance. Design is also what connects sensory affordances to cognitive and physical affordances, so they can be seen or heard or felt (and eventually tasted or smelled) to be used.

Table 1 contains a summary of these affordance types and their roles in interaction design.

Table 1. Summary of affordance types

Affordance Type	Description	Example
Cognitive affordance	Design feature that helps users in knowing something	A button label that helps users know what will happen if they click on it
Physical affordance	Design feature that helps users in doing a physical action in the interface	A button that is large enough so that users can click on it accurately
Sensory affordance	Design feature that helps users sense something (especially cognitive affordances and physical affordances)	A label font size large enough to read easily
Functional affordance	Design feature that helps users accomplish work (i.e., the usefulness of a system function)	The internal system ability to sort a series of numbers (invoked by users clicking on the Sort button)

3.4.1 A positive association of affordance roles as structured HCI design guidance

A design methodology based solely on affordance concepts cannot substitute for an effective design methodology set in a complete development life cycle [Hix & Hartson, 1993; Mayhew, 1999; Rosson & Carroll, 2002]. However, we do offer some guidelines to urge designers to think about how these four kinds of affordance work together naturally in the design of a user interface (or other) artefact.

McGrenere & Ho (Section 6.4 of [2000]) also allude to the possibility of affordances as a framework for design. But they, too, fall short of prescribing a design methodology based on affordances. It is plausible to codify and integrate affordance concepts so that they can be brought to bear systematically in interaction design, but the resulting approach would have to be evaluated in a summative study before one could make claims about the efficacy of this approach as a ‘method’. Nonetheless, it is incumbent on HCI theory to find useful application to HCI design and analysis.

In the case of affordances, the theory offers a way to tie the different kinds of affordance to the HCI (or any human-machine interaction) design process in an organised way. HCI design must address (at least) two components, tasks and artefacts [Carroll, Kellogg, & Rosson, 1991]. For developing the work flow of an application, task analysis is useful to inventory the tasks, and usage scenarios [Rosson & Carroll, 2002] are necessary to guide design. Affordance theory can guide design of HCI artefacts. Each kind of affordance plays a different role in the design of different attributes of the same artefact, including design of appearance, content, and manipulation characteristics to match users’ needs, respectively, in the sensory, cognitive, and physical actions they make as they progress through the cycle of actions during task performance. As Gaver [1991, p.81] says, thinking of affordances in

terms of design roles ‘allows us to consider affordances as properties that can be designed and analysed in their own terms.’ Additionally, even though the four affordance roles must be considered together in an integrated view of artefact design, these words from Gaver speak to the need to distinguish individually identifiable affordance roles.

As an example of how the concepts might guide designers, suppose the need arises in an interaction design for a button to give the user access to a certain application feature or functionality. The designer would do well to begin by asking if the intended functionality, the functional affordance, is appropriate and useful to the user. Further interaction design questions are moot until this is resolved positively.

The designer is then guided to support cognitive affordance in the button design, to advertise the purpose of the button by ensuring, for example, that its meaning (in terms of a task-oriented view of its underlying functionality) is clearly, unambiguously, and completely expressed in the label wording, to help the user know when it is appropriate to click on the button while performing a task. Then, the designer is asked to consider sensory affordance in support of cognitive affordance in the button design, requiring an appropriate label font size and colour contrast, for example, to help the user discern the label text to read it.

The designer is next led to consider how physical affordance is to be supported in the button design. For example, the designer should ensure that the button is large enough to click on it easily to accomplish a step in a task. Designers should try to locate the button near other artefacts used in the same and related tasks, to minimize mouse movement between task actions. Finally, the designer is guided to consider sensory affordance in support of physical affordance in the button design by ensuring that the user notices the button, so it can be

clicked. For example, the button must be a colour, size, and shape that make it noticeable and must be located in the screen layout so that it is near enough to the user's focus of attention. If the artefact is a feedback message, it also requires attention to sensory affordance (e.g., to notice the feedback), cognitive affordance (e.g., to understand what the message says about a system outcome), and physical affordance (e.g., to click on a button to dismiss the message box).

In sum, the concept of affordance does not offer a complete prescriptive approach to interaction design but does suggest the value of considering all four affordance roles together in design of an interaction artefact by asking (not necessarily always in this order):

1. Is the functionality to which this interaction or artefact gives access useful in achieving user goals through task performance (functional affordance, or purpose of physical affordance)?
2. Does the design include clear, understandable cues about how to use the artefact (cognitive affordance), or about system outcomes if the artefact is a feedback message?
3. Can users easily sense the visual (or other) cues about artefact operation (sensory affordance in support of cognitive affordance)?
4. Is the artefact easy to manipulate by all users in the target user classes (physical affordance)?
5. Can users easily sense the artefact for manipulation (sensory affordance in support of physical affordance)?

Considering one affordance role but ignoring another is likely to result in a flawed design.

For example, if the wording for a feedback message is carefully crafted to be clear, complete,

and helpful (good cognitive affordance), but users do not notice the message because it is displayed out of the users' focus of attention (poor sensory affordance) or users cannot read it because the font is too small, the net design is ineffective. A powerful drag and drop mechanism may offer a good physical affordance for opening files, but lack of a sufficient cognitive affordance to show how it works could mean that most users won't use it.

An example of a way that cognitive affordance and physical affordance work together in interaction design can also be seen in the context of designing constraints for error avoidance. 'Graying out' menu items or button labels to show that inappropriate choices are unavailable at a given point within a task is a simple, but effective, error avoidance design technique. This kind of cognitive affordance presents to the user a logical constraint, showing visually that this choice can be eliminated from possibilities being considered at this point. In that sense, the grayed-out label is a cognitive affordance on its own, quite different from the cognitive affordance offered by the label when it is not grayed out.

If cognitive and physical affordances are connected in the design, a grayed-out button or menu choice also indicates a physical constraint in that the physical affordance usually offered by the menu item or button to access corresponding functionality is disabled so that a persistent user who clicks on the grayed-out choice anyway cannot cause harm. Because these two aspects of graying-out work together so well, many people think of them as a single concept, but the connection of these dual aspects is evident to the user interface programmer, who usually must make separate commands or declarations for the cognitive and the physical parts – to gray out the displayed label appearance and to disable the artefact behaviour so it will not respond to a click.

3.4.2 False cognitive affordances misinform and mislead

Because of the power of cognitive affordances to influence users, designers must be aware of their responsibility to use them with caution. When cognitive affordances don't telegraph physical affordances or, worse, when cognitive affordances falsely telegraph physical affordances, users encounter errors. Gibson calls this 'misinformation in affordances'; for example, as conveyed by a glass door that appears to be an opening but doesn't afford passage. Draper and Barton [1993] call these 'affordance bugs'.

Sometimes a door has both a push plate and a pull handle as cognitive affordances in its design. The user sees this combination of cognitive affordances as an indication that either pushing or pulling can operate this as a swinging door. When the door is installed or constrained so that it can swing in only one direction, however, the push plate and pull handle introduce misinformation in the cognitive affordances that interfere with the design as a connection to physical affordances. We know of a door with a push plate and a pull handle that was installed or latched so that it could only be pushed. A 'Push' sign had been added, perhaps to counter the false cognitive affordance of the pull handle. The label, however, was not always enough to overcome the power of the pull handle as a cognitive affordance; we observed some people still grab the handle and attempt to pull the door open.

Another example of a false cognitive affordance showed up in a letter recently received from an insurance company. There was a form at the bottom to fill out and return, with this line appearing just above the form:

----- Do not detach -----

Because that dashed line looked so much like the usual ‘Cut on this line to detach’ cognitive affordance, I almost did detach the form before realizing that the information above, identifying me as a customer, would be lost if I did.

Examples of false cognitive affordances in user interfaces abound. A common example is seen in Web page links that look like buttons, but don’t behave like buttons. The gray background to the links in the top menu bar of a digital library Web site, Figure 5, makes them seem like buttons. A user might click on the background, assuming it is a button, and not get any result. Because the ‘button’ is actually just a hyperlink, it requires clicking exactly on the text.



Figure 5. False cognitive affordances in a menu bar with links that look like buttons

Below-the-fold issues on Web pages can be compounded by having a horizontal line on a page that happens to fall at the bottom of a screen. Users see the line (as a false cognitive affordance) and assume that it is the bottom of the page, and so do not scroll, missing possibly vital information below.

Sometimes a false cognitive affordance arises from deliberate abuse of a shared convention to deceive the user. Some designers of pop-up advertisements ‘booby trap’ the ‘X’ box in the upper right hand corner of the pop-up window, making it a link to launch one or more new pop-ups when users click on the ‘X’, trapping users into seeing more pop-up ads when their intention clearly was to close the window.

McGrenere & Ho [2000] make a point that the case in their Figure 2 labeled ‘false affordances’ is problematic because ‘it is not the affordance that is false; rather, it is the

information that is false' [McGrenere & Ho, 2000, p. 5]. Affordance roles can help clarify this kind of discussion by allowing us to say that a cognitive affordance (the information McGrenere and Ho refer to) can be considered false when it indicates something about a physical affordance that doesn't exist or indicates something incorrect about a physical affordance that does exist.

As another example, I have a radio with a slider switch for selecting between stereo and monaural FM reception, sketched in Figure 6a. The names for the switch positions (Stereo, Mono) are a good match to the user's model, but the arrows showing which way to slide the switch are unnecessary and introduce confusion when combined with the labels.

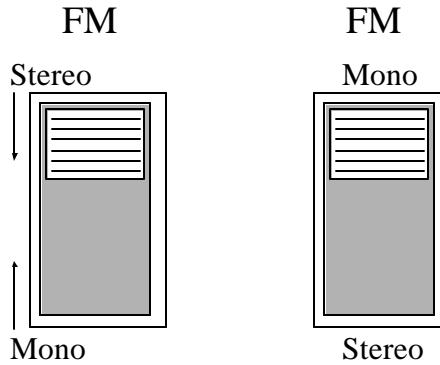


Figure 6. Radio switch with mixed affordances
a. existing design b. better design

The design has mixed cognitive affordances: the names of the modes at the top and bottom of the switch are such a strong cognitive affordance for the user that they conflict with the arrows. The arrows in Figure 6a call for moving the switch up to get monaural reception and down to get stereo. At first glance, however, it looks as though the up position is for stereo (toward the 'stereo' label) and down is for monaural, but the arrows make the meaning exactly the opposite. The names alone, as shown in Figure 6b, are the more normal and natural way to label the switch.

4. The user's role in evaluation and redesign – a trail of user-made artefacts

It is not uncommon to see modifications to designs made by users: trails of user-created artefacts blazed in the wake of spontaneous formative evaluation, boldly taking the original designers to task (and back to a task view). A most common example of trails (literally) of user-made artefacts is seen in the paths worn by people as they walk. Sidewalk designers usually like to make the sidewalk patterns aesthetic – regular, symmetric, and rectilinear. However, the most efficient paths for people getting from one place to the other are often less tidy but more direct. The wear patterns in the grass show where people need or want to walk and, thus, where the sidewalks should have been located. The rare and creative sidewalk designer will wait until seeing the worn paths, employing the user-made artefacts as clues about user needs to drive the design.

As Gaver says, when affordances suggest actions different from the way something is designed, errors are common and signs are necessary. The signs are artefacts, added because the designs themselves did not carry sufficient cognitive affordance. We have all seen the cobbled design modifications to everyday things, such as padding added to prevent bruised knuckles, a better grip taped on, an explanation written on, an important feature highlighted with a circle or a bright colour, a feature (e.g., instructions) moved to a location where it is more likely to be seen. Users add words or pictures to mechanisms to explain how to operate them, enhancing cognitive affordance. Users attach yellow Post-It™ notes to computer monitors and keyboards. A farmer has a larger handle welded onto a tractor implement, enhancing physical affordance of the factory-made handle and its inadequate leverage. A homeowner replaces the street number sign on her house with a larger one, enhancing sensory affordance. Such user-made artefacts are a variation on the ‘user-derived interfaces’

theme of Good, Whiteside, Wixon, and Jones [1984], through which designers, after observing users perform tasks in their own way, modified interaction designs so that the design would have worked for those users.

Figure 7, a photo of a glass door in a convenience store, shows an example of a user-added cognitive affordance. The glass and stainless steel design is elegant: the perfectly symmetric layout and virtually unnoticeable hinges contribute to the uncluttered aesthetic appearance, but these same attributes work against cognitive affordance for its operation.



Figure 7. Glass door with a user-added cognitive affordance (arrow) indicating proper operation

The storeowner noticed many people unsure about which side of the stainless steel bar to push or pull to open the door, often trying the wrong side first. To help his customers with what should have been an easy task in the first place, he glued a bright yellow cardboard arrow to the glass, pointing out the correct place to operate the door.

In this case, the glass had such a strong sensory effect that, although the arrow did add cognitive affordance, it was still a bit difficult to process visually because it looks as though it is ‘floating’ on the glass.

These trails of often inelegant but usually effective artefacts added by frustrated users leave a record of affordance improvements that designers should consider for all their users. Perhaps if designers of the everyday things that Norman discusses [1990] had included usability testing in the field, they would have had the data they needed to accomplish this goal. In the software world, most applications have only very limited capabilities for users to set their preferences. Wouldn’t it be much nicer for software users if they could modify interaction designs as easily as applying a little duct tape, a Post-ItTM, or extra paint here and there?

Figure 8 below shows how a car-owner created an artefact to replace an inadequate physical affordance – a built-in drink holder that was too small and too flimsy for today’s super-sized drinks. During one trip, the user improvised with a shoe, resulting in this interesting example of a user-installed artefact.



Figure 8. A user-made automobile cup-holder artefact (used with permission from Roundel magazine, BMW Car Club of America, Inc. [Howarth, 2002])

Another car example is shown in Figure 9, featuring a car with a rear window having a significantly horizontal orientation. Despite the sporty styling, the design fell short in physical affordance, leading the owner to add an after-market ‘grating’ over the window to ward off reflections from the sun, snow from above, and other material that can too easily accumulate on the flat window.



Figure 9. User-added artefact to make the rear window more usable

As a final example, I occasionally need to use my desktop printer to print a letter on a single sheet of letterhead stationery. Inserting the stationery on top of the existing plain paper supply in the printer does this rather easily. The only problem is that I can't easily determine the correct orientation of the sheet as inserted, which is not obvious to me because:

1. I lack a clear mental model of how the sheet travels through the interior mechanism of the printer,
2. printers can vary in this configuration, and
3. the design of the printer itself gives no cognitive affordance for loading a single sheet of letterhead.

Thus, I have attached my own white adhesive label that says, 'letterhead here, face up and upside down', adding yet another user-created artefact attesting to inadequate design. As

Norman [1990, p.9] says, ‘When simple things need pictures, labels, or instructions, the design has failed.’

5. Applying affordance concepts in usability engineering

The importance of affordance concepts to usability practitioners is in the application to interaction design and evaluation. In our own work at Virginia Tech, we have put these concepts to work within the usability engineering process. We have been working on usability engineering support tools built on a common, theory-based framework called the User Action Framework (UAF), a structured knowledge base of usability concepts and issues [Andre et al., 2001].

5.1. *Adapting Norman’s stages-of-action model*

Norman’s stages of action model of human-computer interaction [1986] had an essential influence on the UAF, along with the cognitive walkthrough [Lewis, Polson, Wharton, & Rieman, 1990], the structure of which is similar in many ways to Norman’s model. Both approaches ask questions about:

- whether the user can determine what to do with the system to achieve a goal in the work domain,
- how to do it in terms of user actions,
- how easily the user can perform the required physical actions, and
- (to a lesser extent in the cognitive walkthrough) how well the user can tell whether the actions were successful in moving toward task completion.

Our work is not the first to use Norman’s model as a basis for usability inspection, classification, or analysis. Several approaches (e.g., [Cuomo & Bowen, 1992; Kaur, Maiden,

& Sutcliffe, 1999; Lim, Benbasat, & Todd, 1996; Rizzo, Marchigiani, & Andreadis, 1997])

have used Norman's model and found it helpful for classifying and communicating about usability problems. Even before the concepts of user-interaction design were stable and well documented in Norman's [1986] model, Rasmussen [1983] provided foundational support by constructing a description of system usage in a functional abstraction hierarchy.

Norman's stages of action model, illustrated in Figure 10, shows a generic sequence of user activity as a user interacts with some machine in the world (annotation outside the box added here).

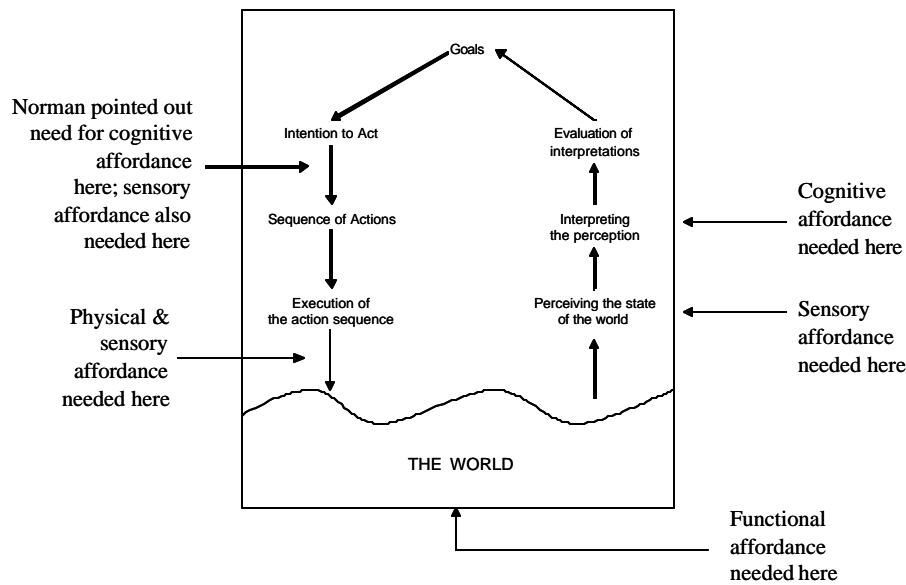


Figure 10. Norman's stages-of-action model (adapted with permission [1990])

Users begin at the top by formulating goals in their work domain. The goals are decomposed into tasks and then into specific intentions, which are mapped to specifications for action sequences. The user then executes the physical actions, causing a state change in the physical world, which is then sensed by the user via feedback, interpreted, and evaluated by

comparing the outcome to the original goals. The interaction is successful if the actions in the cycle so far have brought the user closer to the goals.

Although cognitive affordance can be used to help the user with mental activities anywhere in the top part of Norman's diagram, Norman highlights the essential role cognitive affordance plays on the left-hand side of this model, at the point indicated by our top-most arrow pointing into the figure. This is the point where users map intentions into action sequence specifications prior to making the corresponding physical actions, the point where users most need help in knowing how to do things with a machine/computer. Mismatches between the designer's model and the user's view of this mapping contribute to the well-known Gulf of Execution [Hutchins, Hollan, & Norman, 1986; Norman, 1986]. The most effective way for the interaction designer to help users make this mapping from intention to action specification is with effective design of cognitive affordances (e.g. cues given by labels, icons, and prompt messages).

The right hand side of Figure 10 is where users evaluate their actions by comparing system feedback describing outcomes against their goals and intentions. This is the point where users need the most help in knowing about outcomes. Since system outcomes can be seen only through interaction feedback, mismatches between what designers provide and feedback users need contribute to the well-known Gulf of Evaluation [Hutchins et al., 1986; 1986].

5.2. *From Norman's model to our Interaction Cycle*

As a first step we adapted Norman's model into our Interaction Cycle (see Figure 11), which includes all of Norman's stages but organises them pragmatically in a slightly different way. Like Norman's model, the Interaction Cycle is a picture of how interaction happens for a

human user with any machine, in terms of sequences of cognitive, physical, and sensory user actions.

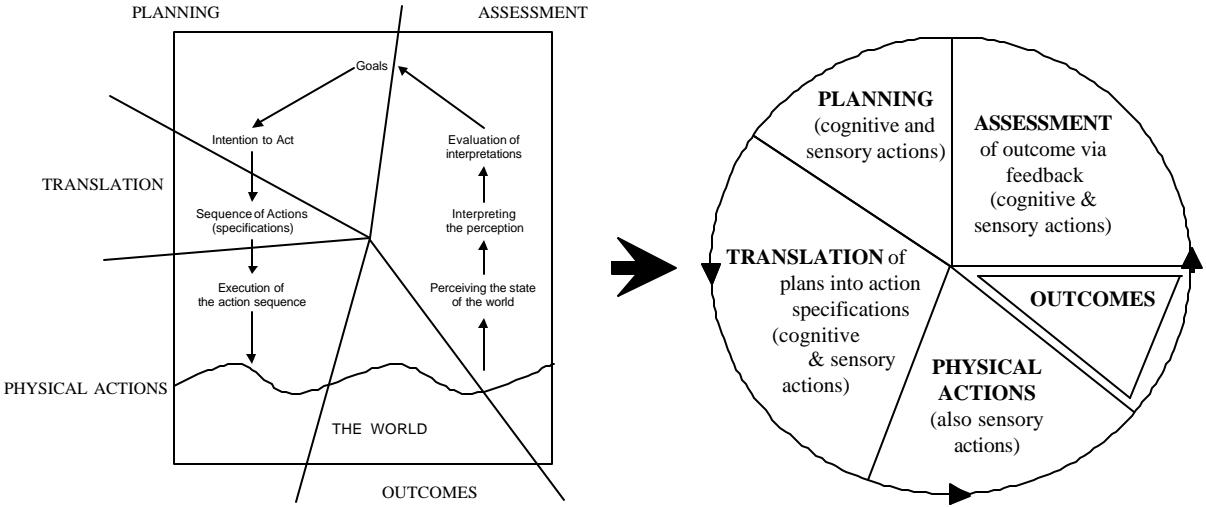


Figure 11. Transition from Norman’s model to our Interaction Cycle

The linear cycle of Planning², Translation, Physical Action, Outcome, and Assessment represents the simplest sequencing, common in a user-initiated turn-taking dialogue style with a computer. Other starting points and orders of sequencing, plus gaps and overlapping, are possible and occur in the world.

The left-hand side of Figure 11 shows how we abstracted Norman’s stages into four basic kinds of user activities, plus Outcomes, to form our Interaction Cycle, on the right-hand side of Figure 11: Planning of actions, Translating task plans and intentions into action specifications, doing Physical Actions, and Assessment of outcomes of those actions. Outcomes in the system occur between Physical Actions and Assessment in what Norman labels ‘The World’. Because the Outcomes category does not include user actions, but is entirely internal to the system and not part of the user interface, we show it as a ‘detached’

² We use capitalization to indicate category names in the User Action Framework.

segment of the Interaction Cycle in Figures 11 and 12. We found that we could associate each observed usability problem and each usability issue, concept, or design guideline with one or more of these categories within the context of a user's cycle of interaction.

5.3. **From the Interaction Cycle to the User Action Framework**

We use the stages of the Interaction Cycle as the high-level organising scheme, as shown in Figure 12 on the right-hand side, for the UAF, a hierarchically structured knowledge base of usability issues, and concepts. The resulting UAF provides a highly reliable [Andre et al., 2001] underlying foundation for usability engineering support tools. High reliability means agreement among users on the meaning of the UAF and how to apply it in the tools.

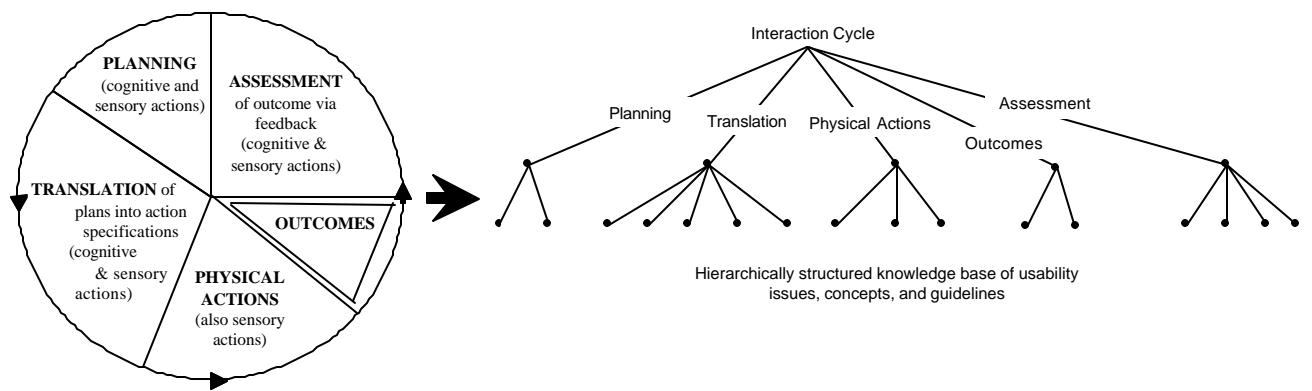


Figure 12. Basic kinds of user actions, plus Outcomes, from the Interaction Cycle as top-level structure of UAF, a usability knowledge base

UAF content under Planning is about how well an interaction design supports the user in determining what to do with the system to achieve work domain goals and includes usability design issues such as the user's model of system, metaphors, and task planning and decomposition. UAF content under Translation is about how well an interaction design supports the user in determining how to do what was planned in terms of user actions on

artefacts in the system, translating task plans into action specifications. Translation includes usability design issues such as the existence of a cognitive affordance (e.g. instructive cue), presentation of a cognitive affordance (sensory issues), content and meaning of a cognitive affordance, and task structure and interaction control (e.g. locus of control, direct manipulation, cognitive directness).

UAF content under the Physical Actions category is about how well an interaction design supports the user in doing the actions. Outcomes represent the system's reaction to physical actions by users, computed by the non-user-interface software. This functionality provides the functional affordances, the usefulness that fulfills the purpose of user actions. Since Outcomes are not directly visible to users, interaction designers must provide feedback representing Outcomes. UAF content under Assessment is about how well feedback in an interaction design supports the user in assessing outcomes of actions.

5.3.1 UAF-based usability engineering support tools

The UAF serves as a common underlying foundation for a suite of usability engineering support tools that we are developing. No tool has its own content; all tools draw on the UAF in a shared relational database for contents of each node in the UAF structure. The mapping to a given tool retains the content and structure of the UAF, but the expression of each concept reflects the specific purpose of the tool. The UAF-based tools include the:

- UAF Explorer tool for teaching usability concepts;
- Usability Problem Diagnosis tool for extracting, analyzing, diagnosing, and reporting usability problems by problem type and by causes;

- Usability DataBase tool for maintaining a life history record of each problem within a project and for supporting aggregate data analysis such as cost-importance analysis [Hix & Hartson, 1993] and usability data visualisation;
- Usability Problem Inspection tool for conducting focused usability inspections, guided by the categories and sub-categories of the UAF; and
- Usability Design Guidelines tool for organising and applying usability design guidelines in a systematic way.

5.3.2 Interaction style and device independence

Norman's stages-of-action model was an ideal starting point for the UAF because:

1. it is a model of sequences of cognitive and physical actions users make when interacting with any kind of machine, and
2. it is general enough to include potentially all interaction styles, platforms, and devices that are likely to be encountered.

The interaction style, platform, and device independence that the UAF derives from its theory base in Norman's model is a long-term advantage. The UAF applies not only to GUI and Web designs, but equally well to 3-D interaction, virtual environments, PDAs, cell phones, refrigerators, ATMs, cars, elevators, and new interaction styles and devices as they arise.

5.4. Affordance roles in the User Action Framework

Affordance is perhaps the single most important overall concept in the UAF, and affordance issues are distributed throughout the interaction design space represented within the UAF.

Cognitive, sensory, and physical actions, each with its own affordance needs in design, often overlap significantly in direct manipulation interaction with computers, in virtual environments, and in non-computer task performance such as in driving a car.

When McGrenere & Ho [2000] use the term ‘degree of affordance’, they are referring to how well an affordance works to help the user, or to the degree of usability afforded. The UAF, via its usability engineering support tools, supports practitioners in their pursuit of high usability, and many of the associated design issues centre on effectiveness of affordances in helping users do things (sensing, cognition, physical actions, and functionality) within the Interaction Cycle. Although all user types need all four kinds of affordances at some time during usage, designs for different kinds of users emphasize different kinds of affordance.

5.4.1 Cognitive affordance in the User Action Framework

Cognitive user actions occur in Planning, Translation, and Assessment within the Interaction Cycle and include a broad range of possibly complex cognitive processes, including rule-based cognition, habitual cognitive actions, explicit causal reasoning for conscious problem solving, and subconscious mental activity. Cognitive affordances appear in the UAF wherever there are issues about helping the user with these cognitive actions, such as knowing what to do (in Planning), knowing how to do it (in Translation), and knowing whether it was successful (in Assessment).

Design quality factors for cognitive affordance (including cues and feedback) are at the heart of a large part of UAF content³, as represented by the sub-categories in Table 2.

Table 2. Representative UAF components relating to cognitive affordance quality

Content, meaning (of a cognitive affordance)
Clarity, precision, predictability of meaning (of cognitive affordance)
Precise use of words
Labels for naming a form field
Labels for buttons, menus
Concise expression
Clearly labeled exits

³ Although very stable, UAF content is subject to on-going refinement and revision to details and wording. Thus, these tables represent a snapshot of UAF categories.

- Completeness and sufficiency of meaning (of cognitive affordance)
 - Complete labels for buttons and menus
 - Complete information for error recovery
 - Complete alternatives in confirmation requests
- Distinguishability (of cognitive affordances)
 - Relevance of content (of cognitive affordance)
 - Convincingness of content, meaning (of cognitive affordance)
 - User-centeredness of wording, design of cognitive affordance content
 - Consistency and compliance of cognitive affordance meaning
 - Error avoidance (in content, meaning of a cognitive affordance)
 - Correctness of content (of cognitive affordance)
 - Make inappropriate options unavailable
 - Anticipate and head-off potential user errors
 - Request user confirmation to avoid potentially costly or destructive errors
 - Distinguish modes
- Layout and grouping (of cognitive affordances)
 - Complexity of layout
- Cognitive directness
 - Direct presentation of cognitive affordance, rather than an encoding
 - Cognitive aspects of manipulable objects, interaction techniques
 - Consistency of manipulation helps user learning
 - Cognitive issues of direct manipulation
 - Direct manipulation paradigm not understood
 - Cognitive affordance content to help know how to manipulate an object, use an interaction technique
 - Mnemonically meaningful cognitive affordances to support human memory limits
 - Content, meaning of cognitive affordances for data entry
 - Appropriate default values for data entry
 - Indicate data type and format expected
 - Field size as indication of allowable data value length
 - Monospace type font (fixed width characters)
 - Meaning contained in cognitive affordance presentation features
 - Preferences and efficiency for content (meaning) of cognitive affordances
 - User ability to set preferences, parameters
 - Accommodating different user classes
 - Style of cognitive affordance content
 - Aesthetics, taste
 - Wording, word choice, vocabulary
 - Anthropomorphism, poor attempts at humor
 - User-centeredness in wording, design
 - Apparent loss of user control due to wording
 - Writing style, reading level (of prompt content)
 - Getting started in a task

An example of a cognitive affordance for Translation is a button label or a menu choice.

During Translation of intentions into action specifications, designers must ask (per Table 2)

if the choice of label wording, for example, is precise enough to provide critical clues required for its proper operation. Is the wording complete enough to avoid ambiguity about the functionality behind a button? Is the wording distinguishable from other choices and

consistent enough to avoid erroneous user actions? Similarly, an example of a cognitive affordance issue for Assessment is the clarity of wording in a feedback message, affecting how well it informs users about errors occurring as the result of certain Physical Actions.

Mnemonic affordances, affordances that help users remember (supporting human memory limitations), are a kind of cognitive affordance. Similarly, time affordances [Conn, 1995], affordances to help users know about or understand time delays in feedback and other output, are a kind of cognitive affordance to support Assessment.

Cognitive affordances are the most abundant type of affordance in interaction designs and account for the most UAF content. Three out of the four major categories of user actions (Planning, Translation, and Assessment) involve cognitive actions. Depending on work domains and user classes, cognitive affordance arguably has the broadest and most important role of all the affordance types in interaction design and, consequently, in the UAF. This is because cognitive affordance is the primary mechanism to support learning and remembering by all users except expert (error-free) users, who have automated Translation actions by training and experience. While expert users may account for a significant percentage of usage time, new or intermediate users comprise the vast majority of the total user population. Even expert users of one system are novice users of many other systems.

We do not report an empirical study in this paper, but our experience from many usability labs in many different settings in business, industry, and government over the years has left a clear impression that flaws in the design of cognitive affordances (or a lack of cognitive affordances) account for as many as 75% of the usability problems observed, primarily in the Translation category of the UAF. Cuomo and Bowen [1992], who also classified usability

problems per Norman's theory of action, similarly found the majority of problems in the action specification stage (our Translation part).

5.4.2 Sensory affordance in the User Action Framework

Sensory user actions occur in support of Planning, Translation, Physical Actions, and Assessment within the Interaction Cycle. For all users except extreme experts, who can make some actions almost 'without looking', each part of the Interaction Cycle generally requires the user to sense (e.g., see, hear, feel) artefacts (including text) in the interaction design that support the corresponding cognitive or physical user activity. Design quality factors for sensory affordance account for significant areas of UAF content, as represented by the categories in Table 3.

Table 3. Representative UAF content about sensory affordance quality

Sensory issues
Noticeability, likeliness to be sensed
Color, contrast
Timing of appearance of cognitive affordance
Layout complexity
Location of cognitive affordance, object with respect to user focus of attention
Focused vs. divided user attention
User focus of attention
Visibility (of cognitive affordance)
Findability
Discernability, recognizability, identifiability, intelligibility (of cognitive affordance)
Legibility of text (of cognitive affordance)
Detectability, distinguishability of sound, force
Bandwidth issues
Sensory disabilities and special limitations
Presentation medium choice (e.g., text vs. voice)
Visual quality of graphics
Auditory quality of audio
Quality of haptic, tactile, force interaction

In Planning, Translation, Physical Actions, and Assessment, UAF issues about sensory affordances are under the Presentation sub-category (presentation, or appearance, of artefacts used as cues, physical affordances, or feedback). As an example, font size or colour used in

button labels and messages might affect text discernability and, therefore, legibility. Sensory issues are separate in the UAF from issues of understanding, which occur under the Content and Meaning category (of both Translation and Assessment).

As an example of discernability, an audio artefact such as a cautionary announcement heard when debarking an escalator, cannot be understood and heeded if the sound is too low in volume or the audio is garbled. As an example of noticeability, a sign in an elevator giving information about the contents of each floor cannot be used to advantage if it is unseen because it is posted too far above eye level. Such cases of difficult Noticeability or Findability might be called: ‘Crouching error, hidden affordance’.

To illustrate sensory affordance in support of physical affordance, clicking on a user interface artefact can be troublesome if the artefact is difficult to see because of poor colour contrast with the background or if it is not noticeable because of poor location (e.g., outside the user’s focus of attention in the screen layout) or timing of appearance (e.g., delayed or not persistent).

An example of a sensory affordance design issue based on a real usability problem case involves a tool palette with a large number of small drawing tool icons in a CAD system. For expert users the icons generally did not present cognitive affordance issues; they usually knew what at least the most frequently used icons meant. But sometimes it proved difficult visually to pick out the needed icon from the dense group in order to click on it. This is a sensory issue in support of Physical Actions for object manipulation, in particular a Findability issue, owing to the overly crowded layout of the visual design. A usability

evaluator might also suspect physical affordance issues here, too, since small size and close proximity might make it more difficult to click quickly and accurately on an icon.

While it is important for designers to help all users see and hear cognitive and physical affordances, special attention is required in design of sensory affordances for users with sensory disabilities. For example, sometimes designers must build in tradeoffs between visual and audio presentation to be selected by users with hearing and seeing disabilities. Issues about sensory disabilities are included in the UAF, extending both Norman's Gulf of Execution and his Gulf of Evaluation [1986] to include sensing.

5.4.3 Physical affordance in the User Action Framework

Well-designed physical affordances support a high level of expert (error-free) user performance and productivity – high usability for power users. Design quality factors for physical affordances, as represented by the categories in Table 4, occur in the Physical Actions category of UAF content, the only category relevant to helping users with physical actions.

Table 4. Representative UAF components of physical affordance quality

Physical Actions (Design helping user do the actions)
Manipulating objects
Physical control
Difficulty manipulating an object (e.g., clicking, grabbing, selecting, dragging)
Object not manipulable, or not in the desired way
Issues about kinesthetics of a device
Issues about manipulating a direct manipulation design
Physical fatigue, stress, strain
Gross motor coordination
Fine motor coordination
Physical layout
Proximity and size of objects as a factor in moving between (Fitts' law issues)
Proximity (closeness) of object as a factor in ability to manipulate reliably
Proximity of objects as a factor in grouping (or sensing of grouping), interference by unrelated objects

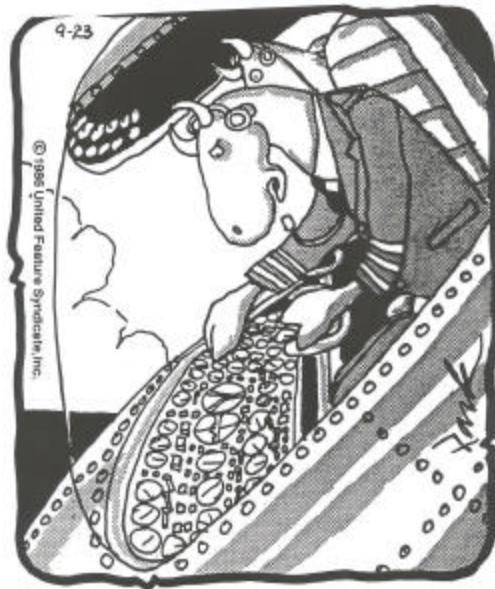
- Display inertia and consistency of object location
- Shape of object(s)
- Inconsistent location of objects
- Physical object design
- Interaction devices, I/O devices
- Inconsistency in the way objects or devices are manipulated
- Interaction techniques, interaction styles
 - Object not manipulable
 - Objects not manipulable in desirable way
 - Physical direct manipulation issues
 - Using direct manipulation when appropriate
 - Preferences and efficiency (for manipulating objects)
 - Efficiency of (single) physical actions (for MOST OR ALL users or user classes)
 - Awkwardness in physical actions for MOST OR ALL users or user classes
 - Accommodating different user classes and physical disabilities
 - Making physical actions efficient for expert users
 - Awkwardness in physical actions for SOME users or user classes

While expert users can ignore many cognitive affordances in an interaction design, all users make use of physical affordances during computer-based task performance. The physical affordance part of the UAF is about operating the ‘doorknobs of the user interface’. Of the two main sub-categories of the Physical Action category in the UAF, sensing artefacts to manipulate and manipulating artefacts, only the latter involves physical affordances. The ‘artefacts’ to be manipulated are the physical affordances for performing tasks. Manipulation issues for physical affordance design include, for example, awkwardness and fatigue, physical disabilities, power performance for experts, and ease of physical clicking as a function of artefact size and distance from where the pointer will be for other related steps in the associated task, according to Fitts’ law [Fitts, 1954; MacKenzie, 1992].

Physical affordance design factors also include the design of I/O devices, direct manipulation issues, physical fatigue, and physical movements associated with virtual environments, gestures, and interaction devices (e.g. different keyboard layouts, haptic devices, speech I/O, and interaction using two hands and feet). Physical affordances are also particularly important to the usability concerns of another kind of user, the disabled user. Extending

Norman's Gulf of Execution [1986] to include Physical Actions, physical affordance issues in the UAF address users with physical disabilities, to whom ordinary designs can pose barriers to physical actions. Disabled users may need assistive technology or accommodation to improve physical affordance to allow, for example, user preferences for larger buttons to support easier clicking by users with limited fine motor control.

The cartoon in Figure 13 is a humorous illustration of a mismatch in physical affordances provided by designers and the physical needs of at least one class of users. Notice, too, the tendency to self-blame by the user, a phenomenon not uncommon in similar situations with computer users.



"Darn these hooves! I hit the wrong switch again!
Who designs these instrument panels, raccoons?"

Figure 13. Mismatch in physical affordances provided by designers and physical needs of users (used with permission from W. B. Park)

A computer-related example of a useful physical affordance for a physical action is the 'snap to grid' feature for precise placement of an object in a drawing program (except when that is

not what the users want, in which case the feature is a hindrance rather than an affordance).

A classic example of a bad system feature with respect to physical affordances is uncontrolled scrolling. In a certain word processor on the PC, dragging selected text to move it outside text showing on the screen causes scrolling when the cursor gets to the top or bottom of the screen. Unfortunately, the speed of scrolling is limited only by the speed of the machine and ends up being too fast for the user to control manually. The result is thoroughly intimidating and frustrating. The system has put the user in a difficult spot, having to hold the mouse button depressed, with the text attached to the cursor, going back and forth unable to find a place to put it.

5.4.4 Functional affordance in the User Action Framework

Effective functional affordance gives all users high usefulness. Design quality factors for functional affordance appear in the Outcomes category of the UAF, the only UAF category containing issues about functionality of the internal, non-user interface software (core application functionality). An example of a functional affordance issue is seen in a case where a word processor performs automatic typing correction, even against the intentions of the user, arbitrarily changing an intended word into an incorrect word. The result for the user is loss of control. This system behaviour definitely affects usability, but it is not just an interaction design problem. Usability engineering developers must work with non-user-interface software engineers to modify this feature, its interface representation and its functionality.

5.4.5 Affordance concepts in usability problem extraction, analysis, and diagnosis

Understanding affordance types and being aware of their roles in interaction design can help practitioners in diagnosing usability problems observed in usability evaluation. As in design, affordances are not the whole story of usability problem analysis. Like design, analysis involving affordance is mostly about analysis of artefacts. The task component must also be analysed by looking at planning support, especially task decomposition, as well as task structure and interaction control (sub-categories under Translation in the UAF).

Usability problem diagnosis begins with observational data, raw usability data often in the form of critical incident observations and verbal protocol, collected in a usability evaluation setting – e.g., lab-based usability testing, usability inspection, or remote usability evaluation. Observational data are converted to complete and accurate usability problem descriptions through problem extraction, analysis, and diagnosis, in which consideration of affordances plays a major role.

As an example, consider the following usability problem from a real-world usability lab.

A user thinks he knows what he is doing on a certain task, but when he selects an object and clicks on an icon, he gets an error message. The user complains that the error message is in a very small font and the colour is too close to the background colour, so he has difficulty reading the message.

Since this case statement is about a message, which is an interaction design artefact, it is appropriate to use affordance concepts to guide the analysis. Questions such as those in Table 5 below (skipping those for Planning in the UAF for now) can help pinpoint the diagnosis:

Table 5. Example affordance-guided problem diagnosis questions

1. Was the trouble in determining which icon to click on (Translation)?
 - a. Was the trouble in seeing the icons, and labels (sensory affordance in support of cognitive affordance)?
 - b. Was the trouble in understanding the meaning of the icons and labels (cognitive affordance in Translation)? Was user confused? Did user make an error?
2. Was the trouble in doing the clicking (Physical Action)?
 - a. Was the trouble in seeing the icon in order to click on it (sensory affordance in support of physical affordance)?
 - b. Was the trouble in doing the clicking quickly, easily, and reliably (physical affordance)?
3. Was the trouble in determining if the outcome of the action was favorable (Assessment) and, if something went wrong, in determining what went wrong?
 - a. Was the trouble in seeing, discerning the feedback message text (sensory affordance in support of cognitive affordance for feedback)?
 - b. Was the trouble in understanding the feedback message content or meaning?

Our example case indicates two possible usability problems. The display of an error message clearly shows that an error must have occurred. When a critical incident arises due to the occurrence of an error and nothing is wrong with the resulting message, the focus is on the error itself and its causes. This is in the Translation (of plans to action specifications) category of the UAF and in question 1 of the table, since this category is about cognitive affordances that help the user determine correctly how to do something and to avoid errors. However, in our current example the user's complaint is about the quality of the message, not the occurrence of the error itself, so we answer 'no' to question 1 in the table for this particular problem. However, the problem of the error occurring is retained and becomes a separate implied problem to be extracted and its diagnosis will require further data (about what happened earlier, probably a cognitive affordance failure, to cause the error).

The physical action of clicking was not an issue, so we answer 'no' to question 2 in the table, but we must answer 'yes' to question 3, which is about feedback and Assessment. An Assessment problem can be about Presentation of feedback (where sensory aspects are found in the UAF, relating to question 3a), including such issues as feedback Noticeability, Discernability, Timing of appearance, and Graphical quality. Or it can be about feedback

Content and meaning (where cognitive aspects are found in the UAF, relating to question 3b), including such issues as feedback Clarity, Completeness, Correctness, and Relevance.

The problem case statement says that the user has difficulty reading the message, which can be ambiguous. An inexperienced practitioner might be tempted to skip further analysis and jump to the conclusion that this about the user not being able to read the error message in the sense of being unable to understand it completely, a common kind of cognitive affordance problem in Assessment. However, the wording of the case statement makes it clear that the problem is about the user's inability to discern the text of the message; the user cannot easily make out the characters in order to read the words. The problem now comes into focus as a sensory affordance problem in the feedback design, found in the UAF under Assessment, Feedback issues, Presentation of feedback, and Sensory issues of feedback. The problem diagnosis is further traced in the UAF to Discernability, and then Legibility of text and then to Font colour and contrast (with background).

Accurate diagnosis is essential to fixing causes of the right problem, the problem that actually affected the user. Different problems, involving different types of affordance, require entirely different solutions (e.g., changing the font size vs. changing the message wording). It is important for practitioners and developers to understand the distinctions, which are often best understood in terms of affordance concepts. Fixing the wrong problem can waste resources and leave the original problem unsolved.

Not fixing all the problems can lead to missed opportunities. For example, improving only the cognitive affordance to avoid the error should make this problem occur less frequently, but would leave the error message problem unsolved for those times when the error does

occur. Revising the expression of the meaning in the error message might be an improvement, but would not solve this sensory affordance problem.

Finally, data visualisation based on affordance types can be used to improve a usability engineering process. This kind of usability data visualisation requires storing records of usability problems for a project in a database with affordance-related attributes. We use our UAF-based Usability DataBase tool, within which each usability problem is stored, having been diagnosed by problem type and causes among UAF categories. We then tag nodes of the UAF with their associations to each affordance type and are able to visualise the usability data as clustered by affordance type. While the interpretation of clusters is an open question, a large number of usability problems involving the meaning of cognitive affordances would seem to imply design shortcomings involving precise use of words, semantics, and meanings of words and icons – shortcomings that might be addressed by hiring a professional writer, for example, to the interaction development team.

Similarly, large numbers of problems involving physical affordances are a possible indicator of design problems that could be addressed by hiring an expert in ergonomics, human factors engineering, and physical device design. Finally, large numbers of problems involving sensory affordances might be addressed by hiring a graphic designer or layout artist. Formal studies will be required to validate the hypotheses behind these expectations.

6. Conclusion and future work

We agree with Norman's concern that the term affordance has been used with more enthusiasm than knowledge. Perhaps the concepts associated with affordance are so natural and so necessary that people either couldn't resist implicit, undeclared extensions or they

may have believed that the kind of extensions we propose were already accepted usage. We have proposed and explored the use of the complementary terms, cognitive affordance, physical affordance, sensory affordance, and functional affordance to refer to the corresponding concepts in interaction analysis and design. We think an independent concept of cognitive affordance is equally important as the concept of physical affordance. It is a good match and a parallel to physical affordance and is essential to interaction analysis and design, as Norman himself has pointed out many times. We also think that sensory affordance is necessary to support cognitive and physical affordance throughout the user's Interaction Cycle.

In order to get the most practical utility from the concept of physical affordance, we have proposed that each reference to it by researchers or practitioners appear with a statement of purpose, which should be supported by functional affordance in the non-user interface software. Finally, we have developed the UAF to connect these and other interaction design concepts in the domain of design and analysis for usability.

We hope that the suggestions here will bridge the gap between Norman's concerns about misuse of affordance terminology and the needs of practitioners to use the concepts in a practical way. Now usability researchers and practitioners can refer unambiguously to all four types of affordance in the context of interaction design and analysis.

We have explored the relationship between the affordance types associated with observed usability problems. Practitioners can apply usability case data to identify where affordance issues are involved in flawed designs and produce case studies of how increased attention to affordances can improve interaction design.

Acknowledgments

Many thanks to Donald Norman for helpful comments on an early draft and his encouragement to publish this article and for his permission to use the diagram of his model in Figure 10. Thanks also to Roger Ehrich, for traveling all the way to Austria to bring me a gift of the wine opener in Figure 2 and for pointing out the user-made artefact as automobile cup holder in Figure 8. Thanks to my colleagues in UAF development, Terence Andre, Steven Belz, and Faith McCreary, for their inputs about affordances over the past few years, and to Deborah Hix for reading the manuscript and making useful suggestions. I'm grateful to Tonya Smith-Jackson for reading the manuscript and making several valuable suggestions. In particular I thank Tonya for helping with terminology, especially the term sensory affordance. Similarly, I wish to thank Elizabeth Buie for several insightful and practical discussions about concepts and terminology involved in sensing, perception, and cognition.

I also wish to express my appreciation to John Karat, North American Editor in charge of this paper, and the anonymous BIT reviewers for supporting publication of this as a concept paper in the face of increasing demand for papers on methodologies and empirical studies. Special thanks to Jeff Weinberg for helping me locate the Gaver papers [1991] and for pointing out McGrenere and Ho [2000], two important references on affordance. Thanks also to Steve Belz and Miranda Capra for examples of false affordances.

Finally, all photos were taken with a small consumer-grade digital camera (brand to remain unnamed to protect the guilty) from our Usability Methods Research Laboratory that has its on-off power switch where most cameras have their shutter-release button. On more than one occasion, after struggling with multiple menus to configure the camera setting just right, at the precise moment of capture, this design ‘affordance’ has led to my unintentionally

shutting off the camera. Further, each time I turn on the camera power, the lens telescopes out, knocking the lens cap off onto the ground (or worse). Let's hear it for good design!

References

- Allen, B. G., & Buie, E. (2002). What's in a word? The semantics of usability. *interactions*, *IX* (2), 17-21.
- Andre, T., Hartson, H. R., Belz, S., & McCreary, F. (2001). The user action framework: A reliable foundation for usability engineering support tools. *International Journal of Human-Computer Studies*, *54* (1), 107-136.
- Andre, T. S., Belz, S. M., McCreary, F. A., & Hartson, H. R. (2000). Testing a Framework for Reliable Classification of Usability Problems. In *Proceedings of the Human Factors and Ergonomics Society 44th Annual Meeting*, Human Factors and Ergonomics Society: San Francisco, CA, 573-577.
- Arnheim, R. (1954). *Art and visual perception: A psychology of the creative eye*. Berkeley, CA: University of California Press.
- Carroll, J. M., Kellogg, W. A., & Rosson, M. B. (1991). The task-artifact cycle. In J. M. Carroll (Ed.), *Designing interaction: Psychology at the human-computer interface* (pp. 74-102). Cambridge, UK: Cambridge University Press.
- Conn, A. P. (1995). Time affordances: The time factor in diagnostic usability heuristics. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, ACM Press: New York, 186-193.
- Cuomo, D. L., & Bowen, C. D. (1992). Stages of User Activity Model as a Basis for User-Centered Interface Evaluation In *Proceedings of the Annual Human Factors Society Conference*, Human Factors Society: Santa Monica, 1254-1258.
- Draper, S. W., & Barton, S. B. (1993). Learning by exploration, and affordance bugs. In *Proceedings of the INTERCHI Conference on Human Factors in Computing Systems (Adjunct)*, ACM: New York, 75-76.
- Fitts, P. M. (1954). The Information Capacity of the Human Motor System in Controlling the Amplitude of Movement. *Journal of Experimental Psychology*, *47*, 381-391.
- Gaver, W. W. (1991). Technology affordances. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, ACM Press: New York, 79-84.
- Gibson, J. J. (1977). The Theory of Affordances. In R. E. Shaw & J. Bransford (Eds.), *Perceiving, Acting, and Knowing*. Hillsdale, NJ: Lawrence Erlbaum Associates.
- Gibson, J. J. (1979). *The ecological approach to visual perception*. Boston: Houghton Mifflin Co.
- Good, M., Whiteside, J., Wixon, D., & Jones, S. (1984). Building a user-derived interface. *Communications of the ACM*, *27* (10), 1032-1043.
- Hartson, H. R., Andre, T. S., Williges, R. C., & van Rens, L. (1999). The User Action Framework: A theory-based foundation for inspection and classification of usability problems. In H. Bullinger & J. Ziegler (Eds.), *Human-computer interaction: Ergonomics and user interfaces (Proceedings of the 8th International Conference on Human-Computer Interaction, HCI International '99)* (Vol. 1, pp. 1058-1062). Mahwah, NJ: Lawrence Erlbaum Associates.
- Hix, D., & Hartson, H. R. (1993). *Developing user interfaces: Ensuring usability through product & process*. New York: John Wiley & Sons, Inc.
- Hochberg, J. E. (1964). *Perception*. Englewood Cliffs, NJ: Prentice-Hall.

- Howarth, D. (2002, April). Custom cupholder a shoe-in. *Roundel, BMW Car Club publication*, 10.
- Hutchins, E. L., Hollan, J. D., & Norman, D. A. (1986). Direct manipulation interfaces. In D. A. Norman & S. W. Draper (Eds.), *User centered system design: New perspectives on human-computer interaction* (pp. 87-125). Hillsdale, NJ: Lawrence Erlbaum Associates.
- Kaur, K., Maiden, N., & Sutcliffe, A. (1999). Interacting with virtual environments: An evaluation of a model of interaction. *Interacting with Computers*, 11, 403-426.
- Koffka. (1935). *Principles of gestalt psychology*. New York: Harcourt, Brace & World.
- Landauer, T. K. (1995). *The trouble with computers: Usefulness, usability, and productivity*. Cambridge, MA: The MIT Press.
- Lewis, C., Polson, P., Wharton, C., & Rieman, J. (1990). Testing a walkthrough methodology for theory-based design of walk-up-and-use interfaces. In *Proceedings of the CHI '90 Conference Proceedings*, ACM Press: Seattle, WA, 235-242.
- Lim, K. H., Benbasat, I., & Todd, P. (1996). An experimental investigation of the interactive effects of interface style, instructions, and task familiarity on user performance. *ACM Transactions on Computer-Human Interaction*, 3 (1), 1-37.
- MacKenzie, S. (1992). Fitts' law as a research and design tool in human-computer interaction. *Human-Computer Interaction*, 7, 91-139.
- Mayhew, D. J. (1999). *The Usability Engineering Lifecycle*. San Francisco: Morgan Kaufmann.
- McGrenere, J., & Ho, W. (2000). Affordances: Clarifying and evolving a concept. In *Proceedings of the Graphics Interface 2000*, Canadian Human-Computer Communications Society: Toronto, 179-186.
- Norman, A. D. (1990). *The Design of Everyday Things*. New York: Doubleday.
- Norman, D. A. (1986). Cognitive engineering. In D. A. Norman & S. W. Draper (Eds.), *User centered system design: New perspectives on human-computer interaction* (pp. 31-61). Hillsdale, NJ: Lawrence Erlbaum Associates.
- Norman, D. A. (1988). *The Psychology of Everyday Things*. New York: Basic Books.
- Norman, D. A. (1999, May/June). Affordances, conventions, and design. *Interactions*, 38-42.
- Rasmussen, J. (1983). Skills, rules, knowledge: Signals, signs, and symbols and other distinctions in human performance models. *IEEE Transactions on Systems, Man, and Cybernetics*, 3, 257-267.
- Rizzo, A., Marchigiani, E., & Andreadis, A. (1997). The AVANTI project: Prototyping and evaluation with a cognitive walkthrough based on the Norman's model of action In *Proceedings of the Designing Interactive Systems (DIS '97) Conference Proceedings*, ACM Press: Amsterdam, 305-309.
- Rosson, M. B., & Carroll, J. M. (2002). *Usability Engineering: Scenario-based development of human-computer interaction*. San Francisco: Morgan Kaufman.
- Thimbleby, H. (1990). *User Interface Design*. New York: ACM Press/Addison-Wesley.
- Thimbleby, H. (2002). Symmetry for Successful Interactive Systems. In *Proceedings of the ACM CHI New Zealand*, 1-9.



AFFORDANCE, CONVENTIONS, AND DESIGN

DONALD A. NORMAN

The Nielsen Norman Group
Web: <http://www.jnd.org>
E-mail: don@jnd.org

I was quietly lurking in the background of a CHI-Web discussion, when I lost all reason: I just couldn't take it anymore. "I put an affordance there," a participant would say, "I wonder if the object affords clicking ... " Affordances this, affordances that. And no data, just opinion. Yikes! What had I unleashed upon the world? "No!" I screamed, and out came this article. I don't know if it changed anyone's minds, but it brought the CHI-Web discussion to a halt (not what good list managers want to happen). But then, Steven Pemberton asked me to submit it here. Hope it doesn't stop the discussion again. Mind you, this is not the exact piece I dashed off to CHI-Web: it has been polished and refined: the requirements of print are more demanding than those of e-mail discussions.

Affordances, Constraints, and Conceptual Models

The word *affordance* was coined by the perceptual psychologist J. J. Gibson [1, 2] to refer to the actionable properties between the world and an actor (a person or animal). To Gibson, affordances are relationships. They exist naturally: they do not have to be visible, known, or desirable.

I originally hated the idea: it didn't make sense. I cared about processing mechanisms, and Gibson waved them off as irrelevant. Then Gibson started spending considerable time in La Jolla, and so I was able to argue with him for long hours (both of us relished intellectual arguments). I came to appreciate the concept of affordances, even if I never understood his other concepts, such as "information pickup." He and I disagreed fundamentally about how the mind actually processes perceptual information (that phrase alone would infuriate him).

Turn now to the late 1980s, when I spent a sabbatical at the Applied Psychology Unit in Cambridge, England. My struggles with British water taps, light switches, and doors propelled me to write *The Psychology of Everyday Things* (POET [5]).

A major theme of POET was the attempt to understand how we managed in a world of tens of thousands of objects, many of which we would encounter only once. When you first see something you have never seen before, how do you know what to do? The answer, I decided, was that the required information was in the world: the appearance of the device could provide the critical clues required for its proper operation.

In POET, I argued that understanding how to operate a novel device had three major dimensions: conceptual models, constraints, and affordances. These three concepts have had a mixed reception.

To me, the most important part of a successful design is the underlying conceptual model. This is the hard part of design: formulating an appropriate conceptual model and then assuring that everything else be consistent with it. I see lots of token acceptance of this idea, but far too little serious work. The power of constraints has largely been ignored.

To my great surprise, the concept of affordance was adopted by the design community, especially graphical and industrial design. Alas, yes, the concept has caught on, but not always with complete understanding. My fault: I was really talking about perceived affordances, which are not at all the same as real ones.

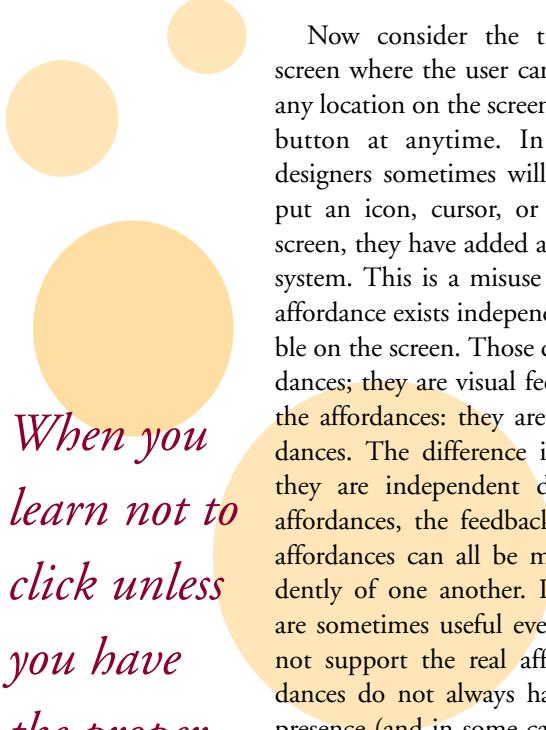
Perceived Affordance

POET was about "perceived affordance." When I get around to revising POET, I will make a global change, replacing all instances of the word "affordance" with the phrase "perceived affordance." The designer cares more about what actions the user perceives to be possible than what is true. Moreover, affordances, both real and perceived, play very different roles in physical products than they do in the world of screen-based products. In the latter case, affordances play a relatively minor role: cultural conventions are much more important. More on that in a moment.

In product design, where one deals with real, physical objects, there can be both real and perceived affordances, and the two sets need not be the same.

In graphical, screen-based interfaces, the designer primarily can control only perceived affordances. The computer system already comes with built-in physical affordances. The computer, with its keyboard, display screen, pointing device, and selection buttons (e.g., mouse buttons) affords pointing, touching, looking, and clicking on every pixel of the screen. Most of this affordance is of little interest for the purpose of the application under design.

Although all screens within reaching distance afford touching, only some can detect the touch and respond to it. Thus, if the display does not have a touch-sensitive screen, the screen still affords touching, but it has no effect on the computer system. While the affordance has useful value in allowing people viewing the same screen to indicate regions of interest, this affordance mainly serves to make the screen-cleaning companies happy: they can sell lots of tissue and cleaning fluid. But this affordance is seldom useful to the interface designer.



*When you
learn not to
click unless
you have
the proper
cursor form,
you are
following a
cultural
constraint.*

Now consider the traditional computer screen where the user can move the cursor to any location on the screen and click the mouse button at anytime. In this circumstance, designers sometimes will say that when they put an icon, cursor, or other target on the screen, they have added an “affordance” to the system. This is a misuse of the concept. The affordance exists independently of what is visible on the screen. Those displays are not affordances; they are visual feedback that advertise the affordances: they are the perceived affordances. The difference is important because they are independent design concepts: the affordances, the feedback, and the perceived affordances can all be manipulated independently of one another. Perceived affordances are sometimes useful even if the system does not support the real affordance. Real affordances do not always have to have a visible presence (and in some cases, it is best to hide the real affordance). And the presence of feedback can dramatically affect the usability and understandability of a system, but quite independently of the affordances or their visibility.

Similarly, it is wrong to claim that the design of a graphical object on the screen “affords clicking.” Sure, you can click on the object, but you can click anywhere. Yes, the object provides a target and it helps the user know where to click and maybe even what to expect in return, but those aren’t affordances, those are conventions, and feedback, and the like. This is what the interface designer should care about: Does the user perceive that clicking on that object is a meaningful, useful action, with a known outcome?

It is possible to change the physical affordances of the screen so that the cursor appears only at spots that are defined to be “clickable.” This would indeed allow a designer to add or subtract the affordance of clicking, much as many computer forms afford the addition of characters only in designated fields. This would be a real use of affordances.

In today’s screen design sometimes the cursor shape changes to indicate the desired action (e.g., the change from arrow to hand shape in a browser), but this is a convention, not an affordance. After all, the user can still click anywhere, whatever the shape of the cur-

sor. Now if we locked the mouse button when the wrong cursor appeared, that would be a real affordance, although somewhat ponderous. The cursor shape is visual information: it is a learned convention. When you learn not to click unless you have the proper cursor form, you are following a cultural constraint.

Far too often I hear graphic designers claim that they have added an affordance to the screen design when they have done nothing of the sort. Usually they mean that some graphical depiction suggests to the user that a certain action is possible. This is not affordance, either real or perceived. Honest, it isn’t. It is a symbolic communication, one that works only if it follows a convention understood by the user.

Constraints and Conventions

When designing a graphical screen layout, designers greatly rely on conventional interpretations of the symbols and placement. Much of the discussion about the use of affordances is really addressing conventions, or what I call cultural constraints. In POET, I introduced the distinctions among three kinds of behavioral constraints: physical, logical, and cultural. These are powerful design tools, so let’s be clear where each is being used.

Physical constraints are closely related to real affordances: For example, it is not possible to move the cursor outside the screen: this is a physical constraint. Locking the mouse button when clicking is not desired would be a physical constraint. Restricting the cursor to exist only in screen locations where its position is meaningful is a physical constraint.

Logical constraints use reasoning to determine the alternatives. Thus, if we ask the user to click on five locations and only four are immediately visible, the person knows, logically, that there is one location off the screen. Logical constraints are valuable in guiding behavior. It is how the user knows to scroll down and see the rest of the page. It is how users know when they have finished a task. By making the fundamental design model visible, users can readily (logically) deduce what actions are required. Logical constraints go hand in hand with a good conceptual model.

Cultural constraints are conventions shared

by a cultural group. The fact that the graphic on the right-hand side of a display is a “scroll bar” and that one should move the cursor to it, hold down a mouse button, and “drag” it downward in order to see objects located below the current visible set (thus causing the image itself to appear to move upwards) is a cultural, learned convention. The choice of action is arbitrary: there is nothing inherent in the devices or design that requires the system to act in this way. The word “arbitrary” does not mean that any random depiction would do equally well: the current choice is an intelligent fit to human cognition, but there are alternative methods that work equally well.

A convention is a constraint in that it prohibits some activities and encourages others. Physical constraints make some actions impossible: there is no way to ignore them. Logical and cultural constraints are weaker in the sense that they can be violated or ignored, but they act as valuable aids to navigating the unknowns and complexities of everyday life. As a result, they are powerful tools for the designer. A convention is a cultural constraint, one that has evolved over time. Conventions are not arbitrary: they evolve, they require a community of practice. They are slow to be adopted and, once adopted, slow to go away. So although the word implies voluntary choice, the reality is that they are real constraints on our behavior. Use them with respect. Violate them only with great risk.

Symbols and constraints are not affordances. They are examples of the use of a shared and visible conceptual model, appropriate feedback, and shared, cultural conventions.

How do you know if the user shares the conventions? Why, with data, of course. This is something that cannot be decided by arguments, logic, or theory. Cultural constraints and conventions are about what people believe and do, and the only way to find out what people do is to go out and watch them—not in the laboratories, not in the usability testing rooms, but in their normal environment.

I still hear far too much dogmatism about what people really “want,” what they “believe,” or how they “really” behave, but I see very little data. It doesn’t take much data.

My partner, Jakob Nielsen, has long argued that you can get these data at a discount: three to five people will give you enough for most purposes [3, 4]. But they need to be real people, doing real activities. Don’t speculate. Don’t argue. Observe.

Concluding Summary

We have many tactics to follow to help people understand how to use our designs. It is important to be clear about the distinctions among them, for they have very different functions and implications. Sloppy thinking about the concepts and tactics often leads to sloppiness in design. And sloppiness in design translates into confusion for users.

In this article I covered the following concepts:

- ◆ The conceptual model
- ◆ Real affordances
- ◆ Perceived affordances
- ◆ Constraints
- ◆ Conventions

The most important design tool is that of coherence and understandability, which comes through an explicit, perceivable conceptual model. Affordances specify the range of possible activities, but affordances are of little use if they are not visible to the users. Hence, the art of the designer is to ensure that the desired, relevant actions are readily perceivable.

Today we do much of our design on computer screens, where the range of possible actions are limited to typing on a keyboard, pointing with a mouse, and clicking on mouse and keyboard switches. Soon we will add spoken words and visual gestures to the list of interactions. All of these actions are abstract and arbitrary compared to the real, physical manipulation of objects, which is where the power of real and perceived affordances lies. Today’s design often lies in the virtual world, where depiction stands in for reality. Many aspects of physical affordances are denied the designer: the alternatives are constraints and conventions. These are powerful when used well. Personally, I believe that our reliance on abstract representations and actions is a mistake and that people would be better served if we would return to control through physical

*Logical and
cultural
constraints
are
powerful
tools for the
designer.*

Don't confuse affordances with conventions.

COPYRIGHT © 1999 DONALD A.
NORMAN. ALL RIGHTS RESERVED.
ACM 1072-5220/99/0500

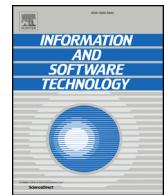
objects, to real knobs, sliders, buttons, to simpler, more concrete objects and actions. But that is a different story for a different time. Moreover, control of our artifacts through abstract commands implemented via typed and spoken items, pointing, and clicking will be with us for a very long time, so we do need to adapt.

Please don't confuse affordances with perceived affordances. Don't confuse affordances with conventions. Affordances reflect the possible relationships among actors and objects: they are properties of the world. Conventions, conversely, are arbitrary, artificial, and learned. Once learned, they help us master the intricacies of daily life, whether they be conventions for courtesy, for writing style, or for operating a word processor. Designers can invent new real and perceived affordances, but they can-

not so readily change established social conventions. Know the difference and exploit that knowledge. Skilled design makes use of all.

References

1. Gibson, J. J. "The Theory of Affordances." In R. E. Shaw & J. Bransford (eds.), *Perceiving, Acting, and Knowing*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1977.
2. Gibson, J. J. *The Ecological Approach to Visual Perception*. Houghton Mifflin, Boston, 1979.
- 3 Nielsen, J. *Usability Engineering*. AP Professional, Boston, 1993.
4. Nielsen, J., and Mack, R. L. (eds.). *Usability Inspection Methods*. John Wiley and Sons, New York, 1994.
5. Norman, D. A. *The Psychology of Everyday Things*. Basic Books, New York, 1988. In paperback as *The Design of Everyday Things*. Doubleday, New York, 1990. ☺



The evolution of agile UXD

Tiago Silva Da Silva^{a,*}, Milene Selbach Silveira^b, Frank Maurer^c, Fábio Fagundes Silveira^a



^a Universidade Federal de São Paulo, Brazil

^b Pontifícia Universidade Católica do Rio Grande do Sul, Brazil

^c University of Calgary, Canada

A B S T R A C T

Context: Agile User eXperience Design (Agile UXD) is a current theme and a trending topic for the future of software development. The integration of UX Design within Agile development is seen as one of the frontiers for Agile Methods as a balance between upfront design as advocated by UX and the you-ain't-gonna-need-it (YAGNI) principle from the agile community must be found.

Objective: In this paper, we analyze the evolution and current state of Agile UXD to provide a brief overview of the topic and to point out still unaddressed gaps, challenges, and future trends.

Method: We systematically analyzed the existing research literature on how this topic evolved over time. We identified three categories with distinctive sets of work and classified them as Early, Middle and Recent years.

Results: We noticed that the Process and Practice dimension has already crossed the line that separates Agile and UXD, the People and Social dimension is crossing this line right now, and the Technology and Artifact is the dimension that took the longest to be addressed, and it did not cross the line yet. Crossing the line means that there is already a full understanding from the Agile side of UX needs and vice versa.

Conclusion: Agile UXD is a need for today's software development teams. However, integrated teams still need to understand that UXD is not a role, but discipline and culture for the whole Agile environment.

1. Introduction

In 2011, during the Agile Manifesto's 10th Anniversary Reunion at the Agile Conference held in Salt Lake City, one of the questions asked was ``What is the next frontier for Agile?'', to which Martin Fowler answered:

“There are two... the integration of operations and the integration with User eXperience (UX) work... I remember, not many years ago, UX people saying: you could not possibly do Agile UX. Everything has to be planned in advance”.

(Agile [1]).

Nowadays, we are living in an experience-driven world: the User Experience (UX) of a software product often determines its success or failure, especially when it faces an end-user market. Therefore, if you are in the software business, you are probably in the UX business (Goethelf and Sneiden, [10]). Also, Agile software development has been characterized differently than plan-based or traditional development methods, mainly with the focus adapting to change and delivering products of high quality through simple work-processes [15].

Henceforth, we are going to address the use of User eXperience

Design (UXD) approaches within Agile processes as Agile UXD.

While Agile focuses on the question of how useful software can be developed, User eXperience Design (UXD) ensures that the goals and needs of the end users are the focus of the development of a product. There is an inherent tension between both schools of thought: agile approaches usually try to reduce and limit upfront analysis and design work while UXD approaches emphasize the need for these. This tension is a core reason why researchers, seeing the value of both arguments, have been investigating how to integrate both approaches.

As delivering highly usable systems is crucial for economic success and for creating business value in a fast-changing environment, the integration of Agile and UXD has been seen as a promising endeavor and has received increasing attention in the last 15 years.

In this paper, we analyze the evolution of Agile UXD – from an academic perspective, examining the literature published in peer-reviewed conferences and journals, and a recent published paper collection [7] on the topic – to provide a brief overview on the topic. We will also highlight still unaddressed gaps, challenges, and future trends. The primary objective here is not to present an extensive analysis of the literature, but, based on a comprehensive study of existing work, provide an overview to students, researchers, and practitioners who intend

* Corresponding author.

E-mail address: silvadasilva@unifesp.br (T.S. Da Silva).

to better understand the evolution of Agile UXD.

2. Research method

To achieve the goals of this paper, we followed the strategy adopted by Brhel et al. [14]. We cross-checked their paper set with our results using the same strings and same databases to identify new milestones from 2012 to 2016. We analyzed the studies that focused on the subject of agile user experience design and classified them along dimensions as described in the next section.

Brhel et al. [14] carried out a literature review following the established guidelines for conducting systematic reviews suggested by Kitchenham [3]. These are a proven means to arrive at a complete and thorough overview of existing research within a domain. We followed their strategy because, to the best of our knowledge, the study presented by Brhel et al. is the latest and most complete literature review on the topic.

Due to our focus – peer reviewed literature – we did not include books like Beyer [8]; Ratcliff and McNeill [12]; Brown [4]; Jongerius et al. [16]; Klein [11]; and Gothelfff and Snieden [10] in our study. However, we acknowledge that some insights are condensed into these textbooks.

The contents of the Cockton et al. [7] book is included in our review because the book is essentially a collection of peer reviewed papers from a NordiCHI workshop in 2014 entitled ‘On the integration of user centred design in agile development’ (Larustottir et al., [13]). Thus, instead of referencing the papers from the workshop, we refer to the extended and improved versions in the book.

To illustrate the evolution within phases, we defined the phases using intervals of five years for each phase to distribute the stages equally. The milestones were chosen based on the first appearance of that topic in a paper. For instance, we do not mention every framework proposed, we just highlight the first one considering the chronological order in the timeline. The reader can find all the framework proposals in the reference list.

3. The evolution of agile UXD over time

Based on the work of Barksdale and McCrickard [9], Brhel et al. [14] defined four dimensions in which the existing literature about the theme is classified: process integration, practice integration, people and social integration, and technology integration. We expanded this classification by adding artifact integration to the technology dimension. Artifact integration represents the incorporation and adaptation of artifacts from both UXD and Agile to mediate teams' communication.

We also merged the processes and practices integration dimensions as well because we consider these aspects strongly related and almost inseparable. As a result, we are using the following dimensions in our analysis:

- *Process and Practice* integration is understood as the merging and synchronizing of UXD and Agile processes, providing a unified process incorporating both perspectives as well as embedding of UXD practices into Agile processes and vice versa.
- *People and Social* integration means changes to the team composition to bring experts from the two different disciplines together as well as the social interaction and the joint creation of knowledge.
- *Technology and Artifact* integration entails the use of technological means to support and coordinate activities as well as the incorporation and combination of artifacts from both processes to mediate communication and create a shared understanding of issues.

Both Figs. 1 and 2 outline the evolution of the field by partitioning it in three periods – Early, Middle, and Recent years. In Fig. 1, we emphasize the first publication addressing a specific topic. This chart

illustrates the topical evolution of the field.

In contrast, Fig. 2 highlights milestones in the evolution by linking topics with papers. In Fig. 1, each dimension is presented for each period in more specific themes, each of which is actually addressed by a paper that is referenced in Fig. 2. In Fig. 2, the milestones – the first time a subject is addressed in a publication – for each dimension in each period are presented.

The timeline in Fig. 2 shows the Agile UXD evolution divided into three Periods – with intervals of five years between each phase: Early, Middle and Recent. In the Early years, the discussion of a topic took started with speculative studies on its importance. During the Middle period, Agile UXD began to identify its particularities in relation to other fields and established its own identity. During the recent years, the community pushed the boundaries and started to face some new limitations to overcome.

4. What IS next for agile ux?

Although Agile UXD is an established research topic, there is always work to be done, issues to be addressed and discoveries to be made.

The overall evidence that we should not manage and control two separate processes is solid. However, we still have a long way to go. Most people admit that while the integration of agile and UXD processes is not always smooth, it is a step forward compared to each of them individually. The adoption of agile and UXD approaches in the industry has grown steadily.

However, there are still open questions and unaddressed gaps.

Currently, there is still a need for new framework proposals to synchronize usability evaluations (UXD perspective) with unit testing or acceptance testing (Agile perspective). Although there are some framework proposals reported in the literature – e.g., Mostafa [6] – agile methods are continually evolving, which requires new ways to integrate UXD concepts. While agile and UXD methods have been combined in several environments, it is essential to develop clear integration guidelines and empirically validate them.

The daily operation of an Agile UXD process is still a concern for developers and designers. They understand the importance of each other's work but still do not know how to make it work on a day-to-day basis. For instance, according to Version One [17], when Agile professionals are asked how to measure progress on a daily basis, customer and/or user satisfaction was only the 7th metric cited by the respondents – behind velocity and iteration, and release burndown.

UXD work tends to be distributed throughout the entire development process, requiring continuous research, continuous design, and continuous evaluation. This implies a need to share the results of UX work with the whole team on an ongoing basis, allowing the team to build a shared understanding. While there are some recent suggestions in the literature to deal with creating a shared understanding between UX designers and developers in agile teams, there are still concerns. The problem of combining UXD and agile methods is an example of a context-dependent issue. Different teams in different contexts use different artifacts and techniques to create a shared understanding.

Another concern is related to the organizational culture. User-Centered Design must mesh with the Agile organizational culture in such a way that everyone in the team will understand UXD as a team discipline rather than a role in the team. A solid understanding of Agile and UXD cultures and practices can help both, developers and designers, to adjust their methodologies, and to adopt techniques that would improve their lines of communication.

This cultural change leads us to face another concern: the future of UXD professionals inside the organization. UXD Specialists have increasingly been working as business analysts as well as coaching development teams to familiarize them with the UX culture.

Distributed teams are a reality and will be increasingly common. Teams will be able to work smoothly with respect to the UX of a software product only with the integration of UXD into the team and the

Agile UXD

Milestones



Early years

Technology/Artifact

- 2004 - Low-fi prototypes
- 2004 - User stories
- 2005 - First mention about artifact-mediated communication

People/Social

- 2002 - Special training and skills are necessary
- 2002 - Close collaboration between designers and developers is necessary
- 2004 - Daily interaction between designers and developers is essential
- 2005 - Usability resources not organized
- 2005 - Mutual understanding and respect are necessary
- 2005 - User research sessions are problematic

Process/Practice

- 2002 - Improving usability does not come without cost or risk
- 2003 - Use of Discount Usability Engineering
- 2005 - First framework to integrate usability practices into an iterative lifecycle
- 2005 - RITE

Middle years

Technology/Artifact

- 2006 - Agile has excessive focus on technology
- 2006 - Paper Prototype is useful in many more ways than just usability testing
- 2006 - UI storyboard
- 2007 - Lightweight prototype
- 2007 - Personas
- 2008 - Extreme personas; Enriched user stories; Personas as tool of communication
- 2008 - UX visions
- 2009 - A research wall to share research findings in daily scrums

People/Social

- 2006 - User involvement, not just customer
- 2007 - Started the discussion about UCD roles
- 2008 - Specialist, Generalist, Generalist/Specialist, U-Scrum Master
- 2009 - Deliver value to customers and users
- 2009 - Collaboration with UX designers
- 2009 - Too much work for UX designers

Process/Practice

- 2006 - 1st framework clearly addressing UCD and Agile methodologies
- 2007 - Parallel tracks, Sprint 0
- 2008 - Traditional usability test did not work well
- 2008 - LDUF and Parallel Tracks

Recent years

Technology/Artifact

- 2010 - Concept mapping, Active story enhanced, LEET
- 2011 - Guidelines, Extreme-Scenario based design
- 2011 - Effectively supporting document and artefacts sharing between physically separated teams
- 2012 - GUITDD
- 2013 - UX issues as acceptance criteria
- 2014 - Focus on artifact-mediated communication

People/Social

- 2010 - Team co-location
- 2011 - Collaborative events
- 2012 - UX designers playing a central role
- 2012 - Integration between designers and developers in a day-to-day basis, engaging with each other
- 2013 - Roles: UX Researcher, Interaction Designer, UI Designer
- 2014 - Influential roles for UX specialists
- 2014 - Customer vs. User
- 2014 - Developers conducting usability evaluations

Process/Practice

- 2010 - Concern with external people
- 2010 - Focus on technology to solve problems
- 2010 - Distributed teams
- 2011 - Impact of Design Thinking
- 2011 - Collaborative Discovery
- 2012 - Integration on a day-to-day basis
- 2013 - RIDE; Patterns for the integration
- 2014 - Level of precision in Up Front Design

Fig. 1. Agile UXD over time and its milestones.

company culture. UXDD (User eXperience Driven Development) [5] may become more widely used. The main idea of UXDD is that, before you get into coding mode, you have customers sign off on wireframes and storyboards for each task offered through the presentation. Looking back 15 years ago, we would only test software if there was enough time at the end of the process. Today, 34% of agile teams use test-driven development, moving test-automation to the forefront of the development process (Version One, [17]).

With a possible UXDD, we need to be innovative in building tools that enable us to perform this user experience driven development. InVision & Marvel are moving in this direction. Another gap is the use of software analytics approaches both to gather requirements and to analyze usage data for making improvements. HotJar is an example of this approach.

Due to distributed teams, there is already a need for digital artifact-mediated communication. This necessity takes us to the need for tools that enable the integration of UI development, as there is in the context of continuous integration. Analyzing the report concerning tools provided by Version One [17], the “customer idea management tool” is the least used one. Why is this the case? For a tool to be seen as useful, it is

necessary that the teams should see value in using them and find them suitable to their work practices. For instance, relying on TDD would not be possible without a good support tool.

The development of computer-assisted usability engineering tools has been discussed for quite some time and there is still no agreement on which the best tools are. Teams tend to appropriate existing tools and choose tools that fit their circumstances.

Tools are needed to support developers in acquiring and sharing UXD and software engineering best practices. They should also be flexible enough for developers to fit them into their particular project context. As said by Seffah and Metzker [2], Agile UXD will be considered more seriously at large if and only if a computer-assisted usability engineering platform is available.

In a long-term, we believe that Agile UXD will be core to the software development culture just as Agile is today. Both processes will be fully integrated.

Agile UXD will be a standard followed by teams that develop interactive software so that they will develop bearing users in mind. There will be infrastructure available for the integration of Agile UXD with operations, which will allow continuous delivery of positive

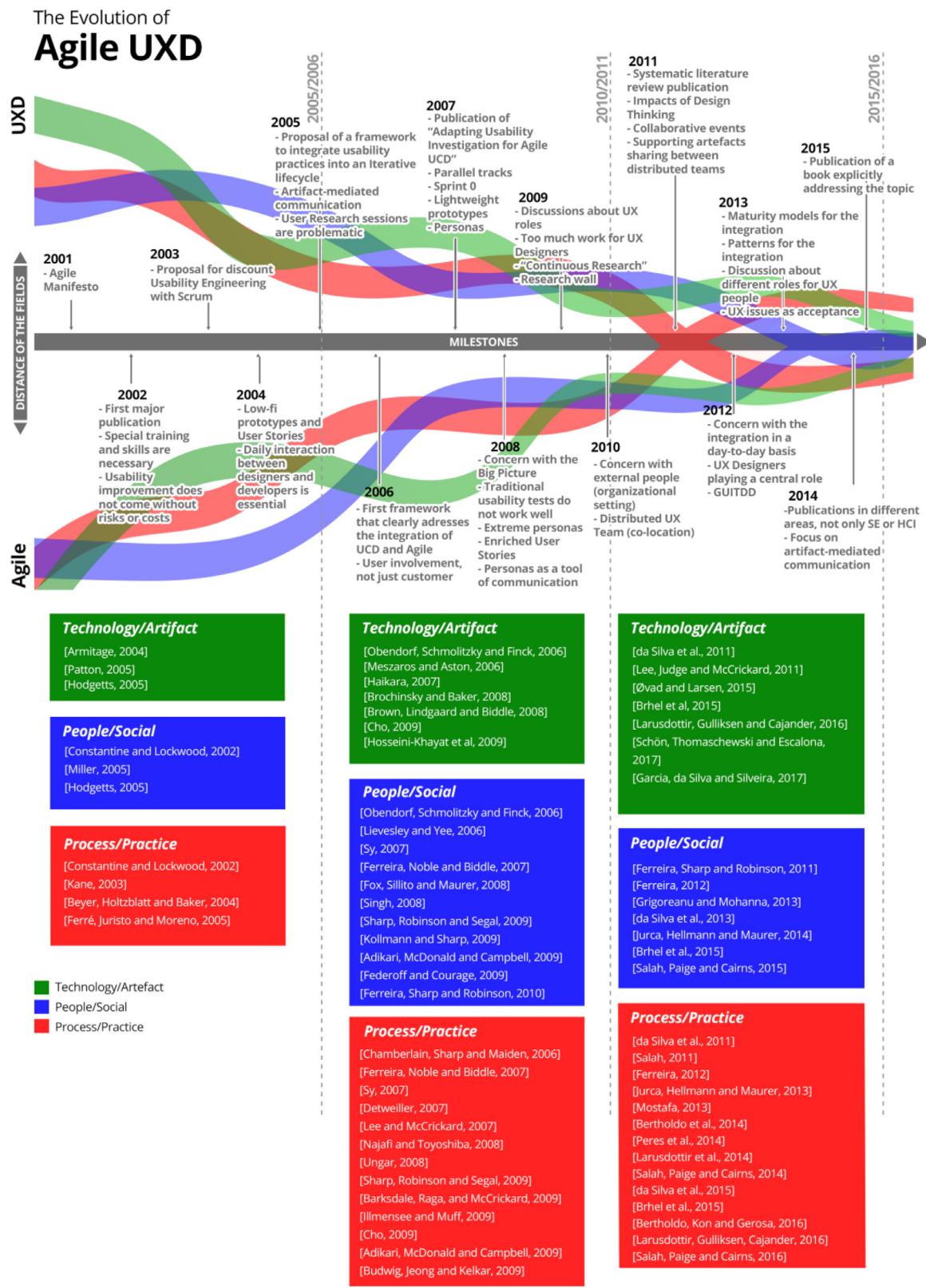


Fig. 2. Agile UXD over time, its milestones and references.

experiences to the end user.

Perhaps, we will be able to predict expected user experiences based on data from previous experiences of groups of users – for example, data from a group of people that represents a particular persona – with the interactive computing system.

5. Final remarks

Work on the Process & Practices dimension (Red Line in Fig. 2) already began in 2002 – based on our conceptual drawing presented in Fig. 2. However, in the beginning, it was just a union of practices from

both approaches without any adaptation. Nowadays, Process and Practice have crossed the line that separates Agile and UXD since the understanding that we cannot have two separate processes is clear.

In 2002, UXD and Agile were far apart regarding the People and Social dimension (Blue line) and now, due to the focus on people and cultural changes, we believe that teams are crossing the line that separates the two fields.

Finally, Technology and Artifact (Green line) is the dimension that took the longest to be addressed. Commercial grade tools that address this dimension are still missing and there is still a way to go before we can achieve an integration.

Lastly, based on an understanding and extensive analysis of the academic literature published so far, we presented an overview of the field and we argue that the topic has reached such a maturity that discussions about its importance are no longer necessary. The goal of the Next stage – to where we are going after the Recent years – then becomes not to create a legacy as much as to simply make sure that the legacy lasts.

If we really want to make Agile User eXperience Design cross the line entirely, we need to really understand users and they must not only be well represented, but also be a real part of the process, regardless of context.

In conclusion, we believe that this short paper may benefit academics in a broader way, due to its wide landscape, as well as students who think that just XP, Scrum, and/or Kanban are enough for all their problems, and industrial readers who will find several references that represent the current body of knowledge.

Acknowledgments

The authors would like to thank FAPESP and the anonymous reviewers whose feedback immensely helped to improve the paper.

References

- [1] A. Alliance, Event: The Agile Manifesto 10th Anniversary Reunion, (2011) August 2011)Retrieved July 10, 2015 from <http://www.agilealliance.org/resources/learning-center/event-the-agile-manifesto-10th-anniversary-reunion>.
 - [2] A. Seffah, E. Metzker, The obstacles and myths of usability and software engineering, *Commun. ACM* 47 (12 December 2004) (2004) 71–76 DOI= <http://dx.doi.org/10.1145/1035134.1035136>.
 - [3] B.A. Kitchenham, *Guidelines for Performing Systematic Literature Reviews in Software Engineering*, Keele University, Keele, UK, 2007 EBSE Technical Report EBSE-2007-012007.
 - [4] D.D. Brown, *Agile User Experience Design: A Practitioner's Guide to Making It Work* 1 Morgan Kauffmann, 2013, p. 256.
 - [5] D. Esposito, *Modern Web Development: Understanding Domains, Technologies, and User Experience* 1 Microsoft Press, 2016, p. 448.
 - [6] D. Mostafa, Maturity Models in the Context of Integrating Agile Development Processes and User Centred Design, Ph.D thesis University of York, 2013.
 - [7] G. Cockton, M. Lárusdóttir, P. Gregory, Å. Cajander, *Integrating User-Centred Design in Agile Development*, Human-Computer Interaction Series 1 Springer International Publishing, 2016, p. 276.
 - [8] H. Beyer, *User-centered agile methods*, Synthesis Lectures on Human-Centered Informatics 1 Morgan & Claypool Publishers, 2010, p. 80.
 - [9] J.T. Barksdale, D.S. McCrickard, Software product innovation in agile usability teams: an analytical framework of social capital, network governance, and usability knowledge management, *Int. J. Agile Extreme Softw. Dev.* 1 (1 June 2012) (2012) 152–177.
 - [10] J. Gothelfff, J. Snieden, *Lean UX: Designing Great Products with Agile Teams*, second ed., O'Reilly Media, 2016.
 - [11] L. Klein, *UX For Lean Startups: Faster, Smarter User Experience Research and Design* 1 O'Reilly Media, 2013, p. 240.
 - [12] L. Ratcliff, M. McNeill, *Agile Experience Design: A Digital Designer's Guide to Agile, Lean, and Continuous (Voices That Matter)*. 1 New Riders, 2011, p. 320.
 - [13] M. Larusdóttir, Å. Cajander, J. Gulliksen, G. Cockton, P. Gregory, D. Salah, On the integration of user centred design in agile development, *Proceedings of the 8th Nordic Conference on Human-Computer Interaction: Fun, Fast, Foundational (NordiCHI '14)*, New York, NY, USA, ACM, 2014, pp. 817–820 <https://doi.org/10.1145/2639189.2654836>.
 - [14] M. Brhel, H. Meth, A. Maedche, K. Werder, Exploring principles of user-centered agile software development: A literature review, *Inf. Softw. Technol.* 61, 1 (May 2015) (2015) 163–181.
 - [15] T. Dingsøyr, T. Dybå, N.B. Moe, *Agile Software Development: Current Research and Future Directions* 1 Springer-Verlag, Berlin Heidelberg, 2010, p. 238.
 - [16] P. Jongerius, A. Offermans, A. Vanhoucke, P. Sanwika, J. van Geel, *Get Agile!: Scrum for UX, Design & Development* 1 BIS Publishers, 2013, p. 144.
 - [17] Version One. 2015. State of Agile Survey 2014. Agile Made Easier. Version One. 116; pages.
- Tiago Silva da Silva** received his M.Sc. from Pontifical Catholic University of Rio Grande do Sul in 2008 as well as his Ph.D. in Computer Science in 2012. He was a visiting researcher at University of Calgary in 2010 and 2011. He is currently an Assistant Professor at the Science and Technology Institute of Federal University of São Paulo (UNIFESP), doing research on agile software development, user-centered design, integration of agile and user-centered design, software fault and usability prediction, and evidence-based software engineering.
- Milene Silveira** received her M.Sc. in Computer Science from Federal University of Rio Grande do Sul in 1996 and her Ph.D. in the same area from Pontifical Catholic University of Rio de Janeiro in 2002. She is a Professor at Pontifical Catholic University of Rio Grande do Sul since 1994, doing research on human-computer interaction, focusing on its relations with agile software development, data visualization, end-user development, and social networks.
- Frank Maurer** is a professor of computer science. He was one of the first researchers investigating agile methods and is widely published in the area. His current research interests are Agile methods, multi-surface systems, and engineering immersive analytics applications. He was program chair and program committee member of a substantial number of agile conferences over the last 15 years.
- Fábio Silveira** received his M.Sc. from Federal University of Rio Grande do Sul in 2001 and his Ph.D. in Computer Science from Technological Institute of Aeronautics (ITA) in 2007. He was a visiting researcher at Technische Universität Berlin from June to August 2009. He is currently an Associate Professor at Science and Technology Institute of Federal University of São Paulo (UNIFESP), doing research on object-oriented and aspect-oriented software testing, experimental software engineering, agile methods, and metadata.

Integrating User Experience into Agile

An Experience Report on Lean UX and Scrum

Manal M. Alhammad

Department of Software Engineering
King Saud University
Riyadh, Saudi Arabia
manalhammad@ksu.edu.sa

Ana M. Moreno

School of Computer Science
Universidad Politécnica de Madrid
Madrid, Spain
ammoreno@fi.upm.es

ABSTRACT

The integration of Agile development and user experience (UX) has received significant attention over the past decade. The literature contains several process models and wide-ranging discussion about the benefits and challenges of this integration. However, academia has given this integration short shrift. In fact, there are very few publications covering educational courses dealing with Agile development and UX. In this paper, we report on our experience of designing and running a graduate software engineering course that covers the integration of Lean UX into Scrum and employs gamification to improve student engagement. We identified six lessons learned that new point to important aspects to be considered when integrating Agile development and UX in academia. For example, we discuss the complexity of managing Lean UX activities in short sprints or how the design of a testable and tactical hypothesis can be one of the most challenging aspects of the Lean UX process.

CCS CONCEPTS

- **Software and its engineering → Software creation and management → Software development process management → Software development methods → Agile software development**
- **Human-centered computing → Interaction design → Interaction design process and methods → User centered design**
- **Human-centered computing → Interaction design → Interaction design → Interaction design theory, concepts and paradigms**

KEYWORDS

Software engineering education, Agile, User experience, Gamification

ACM Reference format:

Manal Alhammad and Ana Moreno. 2022. Integrating User Experience into Agile: An Experience Report on Lean UX and Scrum. In 44th International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET '22), May 21–29, 2022, Pittsburgh, PA, USA. ACM, New York, NY, USA, 12 pages. <https://doi.org/10.1145/3510456.3514156>

1 Introduction

While efforts in the integration of Agile and UX practices (Agile UX) goes back a long time [1, 2], the field has received significant attention over the past years in the software industry, pushing forward the frontiers of Agile methods [3, 4]. Research efforts in this field aim to combine the two processes based on the premise that Agile and UX Design (UXD) share common principles, such as iterative development, emphasis on the user, and team coherence. This integration process is still an open issue, because, as discussed later, different challenges and gaps have been identified [3, 5].

From an academic point of view, little is known so far about this integration [6]. On one hand, even though Agile practices are extensively covered by academic courses, very few SE programs include usability [5] and, when they do, core UXD principles and practices are not always properly addressed [6–9]. On the other hand, very few publications have, to the best of our knowledge, specifically reported academic courses that cover the integration of Agile and UX. We believe that the SE education community has a very important role to play in addressing the challenges of Agile UX in industry by educating future software engineers in UXD principles and processes and by highlighting the value of adopting UXD throughout the Agile development process. This is a must in order to introduce this new culture into organizations following a bottom-up approach.

In this paper, we describe a two-year experience in teaching a graduate SE course, designed to introduce students to an integrated approach to developing Agile projects using UX principles and practices. Particularly, we address the integration of Scrum and Lean UX. To do this, we employ a newly developed framework called GLUX (Gamified Lean UX) [10]. GLUX guides the integrated development process using gamification to motivate agile teams to adopt Lean UX practices.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ICSE-SEET '22, May 21–29, 2022, Pittsburgh, PA, USA

© 2022 Copyright is held by the owner/author(s). Publication rights licensed to ACM.

ACM 978-1-4503-9225-9/22/05...\$15.00
<https://doi.org/10.1145/3510456.3514156>

The aim of this paper is to provide useful insights and suggestions for SE educators on how to design and manage the integration of Lean UX with Scrum in an educational context. However, most of the resulting insights may also be applicable in industrial contexts under particular conditions. Generally, beyond contributing to SE education literature by extending existing Agile and UX teaching experiences, we also hope that this paper will contribute to the discussion within the SE community in order to achieve a better understanding of the current trends and challenges of Agile and UX integration.

2 Background

2.1 Recent Developments in Agile and UX Integration

The integration of UX and Agile software development has been addressed extensively in the literature, leading to the proposal of new methods and techniques for merging the two domains, as well as discussions of the challenges of such integration. Research by Da Silva et al. [3] and Curcio et al. [5] presents a comprehensive and high-level overview of the field, providing an understanding of the evolution of Agile and UX over the past two decades and identifying the different strategies, forms, and challenges of integrating both disciplines. However, most of the integrated processes and techniques published in the literature are rarely followed in practice [3, 10]. The proposed solutions often do not provide a comprehensive integration process that takes into account the organization's culture [12] or team collaboration and communication issues [13]. Nor do they address the lack of basic UX knowledge among software developers, which can be an obstacle to collaboration with UX professionals [11]. Additionally, there are very few empirical studies that evaluate the proposed approaches in industrial and educational contexts. Da Silva et al. [3] stress the need not only to develop guidelines for the integration process, but also to empirically validate those guidelines and approaches. Jurca, Hellmann and Maurer [14] and Curcio et al. [5] confirm that there are few rigorously conducted studies on Agile UX and that most results are inconclusive with respect to the applicability and generalizability of the proposed frameworks in contexts other than the one for which the framework was designed. Wale-Kolade [15] and Bruun and Stage [16] argue that training agile teams in UX methods is essential to address the integration and bridge the gap between software developers and UX professionals.

However, to the best of our knowledge, there are only two studies that discuss the integration of Agile and UX from a training perspective. Péraire [6] presents a Carnegie Mellon University graduate-level course that focuses on integrating interaction design and requirements engineering in the context of dual-track agile, following a mixed approach combining flipped-classroom and traditional delivery. She shares her experience in designing and teaching the course over a four-year period (2015-2019) and discusses the challenges involved in teaching such a course. The other published paper was by Felker, Slamova, and Davis [17],

where they report on their experience in integrating UX design with Scrum in the context of a summer research project for undergraduate students. The approach to the integration of UX into Scrum in that study is based on the ad-hoc adoption of a few UX practices, such as contextual inquiry, prototyping, and formative UX evaluation techniques. They also discuss the challenges they faced with this experience and outline a few lessons learned in the form of practical suggestions and guidelines for educators and practitioners when integrating UX into agile. Our paper extends the work of Péraire [6] and Felker, Slamova, and Davis [17] by reporting our experience in designing and teaching a graduate-level course that covers the integration of Agile and UX. Our work differs from [6] in that we address the integration of Agile and UX into a single process, focusing on Lean UX and Scrum, rather than managing parallel interwoven tracks of design and development as in dual-track agile. On the other hand, our work differs from [17] in that we focus specifically on merging Lean UX with Scrum, following a mixed approach of traditional lectures and project-based learning (PBL). We also adopt a novel course design approach by employing gamification as a motivational technique.

2.2 Previous Work on Gamification in SE Education and Agile

By definition, gamification employs the philosophy and techniques of game design into non-game contexts to induce a target behavior in people and improve their motivation and engagement [18]. Evidenced by the growing number of publications, gamification in SE is seen as a promising technique to improve software engineers' motivation and engagement and to promote SE best practices [19], [20]. Efforts in this field are motivated by the challenging nature of software development which can entail different practices and processes that are considered repetitive and time consuming, such as code review and bug hunting [19]. While empirical research in this field is still limited, early results paint a positive-leaning picture of the effectiveness of gamification in SE [19, 20]. Most notably, gamification has been implemented in requirements engineering [21], in software project management [22], and in SE education [23]. However, more empirical evidence is necessary to establish reliable conclusions in the SE field [24]. Gamified learning was recognized as a very promising, albeit little explored, approach for improving the learning process and outcomes in SE education [7].

The literature also offers three systematic mappings of gamification applied in SE education [23], Agile [25], and Software Process Improvement [26]. Those studies complement existing research on gamification applied to SE and explore to what extent gamification improves or impacts the motivation and engagement of software engineers. Among the main findings, the authors confirmed that gamification is in its early stages in the field of SE, and very few empirical studies exist. This is mostly consistent with the findings of [19, 20], and [24]. The authors also highlight that gamification is context dependent, and it integrates complex aspects such as human behavior and motivational theories, which makes it far from straightforward. In addition, the authors also

raised several critical issues that need to be carefully considered by practitioners who aim to gamify SE processes. For example, gamification should be “integrated into” or “combined with” existing processes or activities; high-intensity gamification, in which the process is overloaded with gamification elements, may have a negative impact, and gamification should not require significant changes that entail a steep learning curve.

3 Our Story

3.1 Why did we choose Lean UX and Scrum? And why did we gamify the integration process?

Agile Methods is a graduate course offered as part of the MS in Software Engineering program at the Universidad Politécnica de Madrid. The aim of this 4-ECTS course is to provide students with an overview of the main techniques used in Agile methods and how they interact with particular quality issues. Before 2019, the course focused on how Agile development processes can account for usability practices to improve the overall quality of the product. This was based on recognizing usability not only as a quality attribute or a non-functional requirement, but also as a crucial factor for delivering successful software products in highly competitive markets [27]. Students were introduced to the main usability heuristics [28], as well as usability guidelines to assure that usability is dealt with from the beginning of the development process [29]. However, students perceived usability as a nice-to-have quality attribute instead of considering it as an essential part of the Agile development philosophy.

In our search for an Agile and UX integration framework for adoption in academia, we discarded the Agile UX approaches published in the scientific literature (journals and conferences), because, as mentioned in Section 2.1, they are not mature enough and, from a practical point of view, are described only briefly in one or two scientific publications where not enough material is provided for an in-depth study and application. We opted instead for Lean UX, which was presented in the book *Lean UX: Designing Great Products with Agile Teams* [30]. Lean UX has a lightweight and iterative nature of the process that is based on design thinking, lean startup, and Agile. As a result, its operation is aligned with Agile development processes.

Additionally, a full and detailed description of the UX practices to be included in the Agile process, as well as specific guidelines and examples on how and when to integrate such practices into a Scrum process are provided in [30]. Consequently, enough suitable material is available. In an attempt to motivate and engage our students to carry out Lean UX activities collaboratively and integrated with Agile, we also employed gamification and designed a framework called GLUX to provide novice Agile teams with effective and engaging help to manage the integration process [10].

3.2 Introducing GLUX

GLUX¹ is illustrated in Figure 1 and provides Agile teams with structured and detailed instructions on how Lean UX tactics can be integrated into Scrum [10]. GLUX is primarily designed for small Scrum teams who are struggling with integrating UX activities into the development process, particularly in the absence of UX specialists. Even if a UX specialist is on hand, the GLUX framework can help align the workflow of the whole team towards building user-centered software products and establish a shared understanding of the product throughout the development process. The GLUX framework is divided into two fundamental parts:

- a. Five Lean UX tactics to integrate into Scrum: Hypotheses, Design Studio, Experiment Stories, Minimum Viable Product (MVP), and Weekly User Experiments.
- b. A customizable gamification strategy based on three game techniques: rewards (points and badges), challenges, and levels.

The key contribution of GLUX is depicted in part b of the GLUX framework. The gamification strategy was based on the lessons learned from three systematic mappings of the literature of gamification applied in SE education, agile, and SPI [23], [25], [26]. Our main goal was to explore how gamification was put into practice in the field of software engineering, and to what extent it improves or impacts the motivation and engagement of software engineers. Accordingly, those lessons were taking into consideration throughout the design process of GLUX since they point to several critical aspects that need to be carefully considered when gamifying software engineering processes.

A typical scenario of an Agile team following GLUX is as follows: the team brainstorms and creates a list of hypotheses concerning their assumptions about the needs of potential users, which they discuss with the product owner (PO) during product backlog refinement. The team kicks off the sprint with a design studio in which the team picks a hypothesis as a theme to guide the work of the upcoming sprint/s and start sketching and discussing design ideas accordingly. Immediately after the design studio, the sprint planning meeting should take place. The team decides which user stories they are going to develop and plans for the weekly user experiment in which they put to test a version of their developing product, i.e., an MVP. These plans are captured in what is called an experiment story. The main goal of running weekly user experiments is to validate the team’s hypotheses and collect constant feedback to improve the product.

The team should be rewarded for applying each Lean UX tactic and receive additional rewards for doing it collaboratively. The Scrum team is encouraged by special rewards for employing Lean UX tactics for the first time, as well as for rising to some Lean UX challenges.

¹ The full documentation of the GLUX framework is summarized at <https://bit.ly/GLUXICSE>.

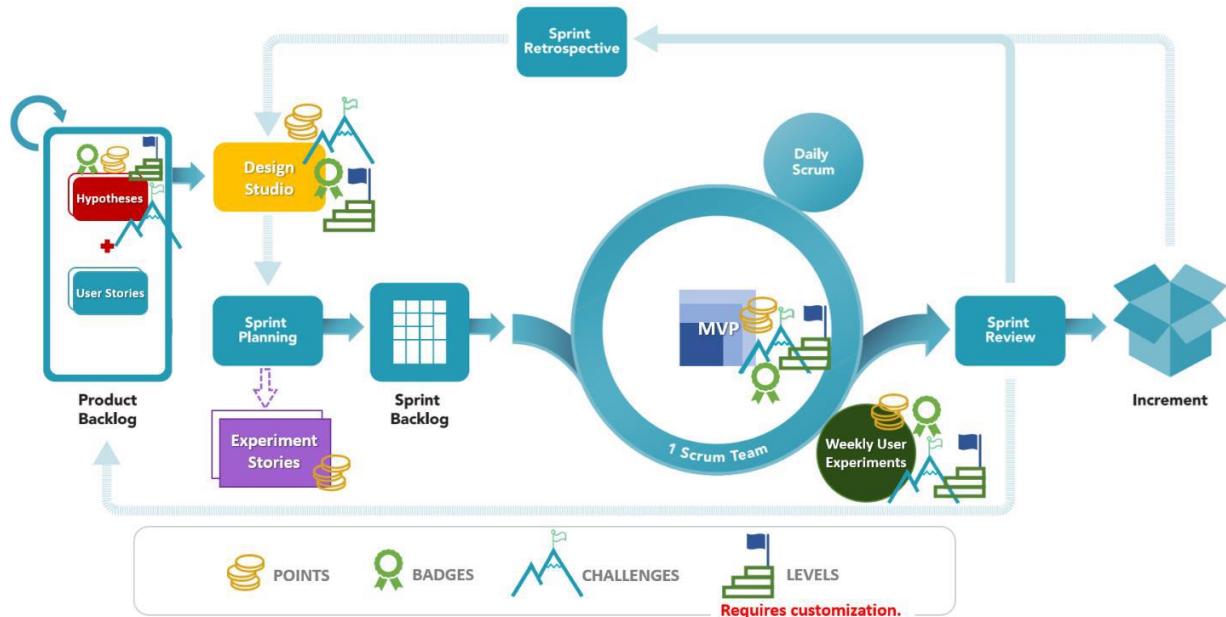


Figure 1. A general overview of the GLUX framework, integrating Scrum with Lean UX using gamification.

Challenges are set by the team based on the current difficulties and issues that the team faces regarding UX. The rewards earned should help the team to move up through levels of expertise in Lean UX and are added to the achievements board. The achievements board is a physical or virtual space, where the collected points, badges, and levels of the team are posted. The aim of this board is to visualize the team's achievements and progress in integrating Lean UX tactics.

3.3 Course Design

This section describes the course structure and study design of two academic semesters (Fall 2019 and Fall 2020) in which GLUX was implemented.

3.3.1 Learning Objectives

The learning objectives of the Agile Methods course are as follows:

- Understand the agile principles, roles, practices, and artifacts used in Scrum.
 - Develop an agile mindset that values early failure, collaboration, continuous learning, continuous improvement, and continuous discovery.
 - Establish basic knowledge of Lean UX principles and tactics.
 - Learn and practice how UX work can be integrated into the workflow of agile development processes.
 - Appreciate the continuous nature of UX and its importance in building high quality software.
- Effectively communicate and collaborate with a team.

3.3.2 Overall structure

The Agile Methods course is divided into three parts. The first part is delivered as traditional lectures where the course instructor presents and explains the main Agile development processes and Scrum concepts. The second part focuses on the Lean UX development philosophy and the integration of Lean UX tactics into Scrum. Additionally, the lectures included a few activities like Scrum City using Legos, Planning Poker, and a hands-on workshop to help the students fully understand key Lean UX Canvas concepts [31]. The third part of the course involved developing a team project following GLUX. The course also includes a retrospective class at the end of the semester in which students and instructors conduct an appraisal of the course and the project development process.

We collected data from students through ([pre](#) and [post](#) experience) questionnaires, unstructured interviews, and the analysis of the students' project reports. The first questionnaire, distributed to the students during the introductory lecture, aimed at performing an initial assessment of the students' Agile and UX knowledge and experience. The second questionnaire was administered during the retrospective class and sought the students' opinions on the process of integrating Scrum and Lean UX. During this last class, we also used unstructured interviews to guide the discussion.

3.3.3 Delivery Format

As mentioned, the course delivery follows a mixed approach, combining traditional lectures with in-class activities, and PBL. In 2019, the lectures and the associated activities were delivered in-class.

Hypothesis	Design Studio	MVP
Rules	Rules	Rules
<p>1. Hypotheses should be written during product backlog refinement and reviewed every sprint.</p> <p>2. Hypotheses writing/review should be done as a teamwork.</p>	<p>1. A UX design studio should be conducted at the beginning of every sprint.</p> <p>2. All team members should participate in sketching and ideating.</p>	<p>1. The team should create an MVP (a working or non-working MVP) before the upcoming weekly user experiment.</p>
Scoring	Scoring	Scoring
<p>Writing or reviewing hypotheses for each sprint → 5 points!</p> <p>The participation of all team members → 10 points!</p> <p>Brainstorming hypotheses during first backlog refinement → The "Thinkers" badge!</p>	<p>Running a design studio each sprint → 5 points!</p> <p>The participation of all team members → 10 points!</p> <p>Running a design studio in the first sprint → The "Designers" badge!</p>	<p>Delivering an MVP before the weekly user experiment → 5 points!</p> <p>Delivering a "working" MVP before the weekly user experiment → 10 points!</p> <p>Delivering a working MVP before the end of the first sprint → The "Makers" badge!</p>
Challenge	Challenge	Challenge
<p>Reviewing hypotheses with PO and stakeholders for 3 sprints in a row → The "Philosophers" badge!</p>	<p>Running a design studio for 3 sprints in a row → The "Senior Designers" badge!</p>	<p>Creating an MVP for 3 sprints in a row → The "Builders" badge!</p>

Figure 2. Three examples of GLUX Cheat Cards, providing a visual summary of GLUX's rules and scoring system.

However, the entire course was delivered online in 2020, due to the COVID-19 global pandemic. The main challenge for us during online delivery was the management of in-class activities and the course project, particularly the maintenance of student engagement and interaction. Therefore, we adopted a flipped-classroom approach in 2020, where we videorecorded the lectures for the students to watch before the actual lecture time. This left room for interactive group learning activities, such as open discussions and exercises within the virtual classroom.

3.3.4 Course Project

In the course project, students simulate a real-world agile project. To do this, they develop an application for managing and funding entrepreneur business ideas, applying Scrum and Lean UX techniques. In the Fall 2019 semester, the 17 students enrolled in the course were divided into three teams. In Fall 2020, the 14 students enrolled in the course were also divided into three teams. Each team was required to develop the project within three one-week sprints. Teams are expected to apply the GLUX framework integrating the Lean UX tactics taught in the lectures (Section 0) into the Scrum process (using the guide discussed in Section 0). Upon adopting these tactics, the teams earn and collect points and badges, and are encouraged to carry out some challenging tasks related to Lean UX. Finally, teams are given the freedom to choose the technological ecosystem to be used in developing their course project and the tools and libraries, for example, to implement user interfaces (e.g. Bootstrap, Vue.js, React.js) or perform user testing (e.g. Maze).

3.4 How did our students use GLUX?

As mentioned, the GLUX framework offers a high-level gamification strategy that requires few initializations and customizations, such as assigning numerical values for each Lean

UX tactic and designing team-specific challenges, rewards, and levels.

Accordingly, we specifically designed a GLUX guide², which illustrates a customized version of the GLUX framework to accommodate our context: three agile teams of graduate students developing a software project using an integrated Scrum and Lean UX process over three one-week sprints. The goal of designing the GLUX guide is to provide a structured process for novice agile teams to integrate Scrum with Lean UX. In addition to the Lean UX tactics, the guide offers additional details on some aspects that might be challenging for novice teams. For example, it explains the different options for a team to manage UX work in the product backlog. In the GLUX guide, we also customized the gamification strategy based mainly on the framework's instructions and included GLUX's cheat cards (Figure 2). Cheat cards are essentially a visual summary of each Lean UX tactic in terms of the rules for integrating such tactics in Scrum, how the scoring system works, and one proposed challenge. Cheat cards made it easier for the students to quickly check the instructions on how to integrate each Lean UX tactic and how the gamified part works.

4 FINDINGS AND LESSONS LEARNED

Described in Table 1, this section discusses the six lessons learned we identified from teaching the course for two semesters (Fall-2019 and Fall-2020). They essentially outline the key issues that we came across during the design and delivery of the course (column 1). We also provide a few suggestions on how such issues can be addressed (column 2 and 3). Some of the recommendations are based on our experience, and others were derived from the literature on SE education and Agile UX (in the latter case, the corresponding references are added).

As discussed in section 3.3.2, data collection occurred through two questionnaires (pre and post experience), unstructured

² Available at <https://bit.ly/GLUXGUIDE>

interviews, and the analysis of the students' project reports. We followed the qualitative data analysis approach by [32] to synthesize the six lessons learned presented in this section, using NVivo 12 as a supporting tool. The approach by [32] offers a qualitative data analysis process that is based on Grounded Theory (GT). Unlike GT where codes should not be defined *a priori* and should emerge from the analysis process, the qualitative data analysis approach of [32] starts with "seed categories". Seed categories are an initial set of codes that come from the goals of the study, the research questions, or any predefined area of interest to guide the analysis process. Accordingly, we first formulated a set of "seed codes" that reflect a few aspects we are interested in about the learning process and the overall course material and structure. For example, we were interested to examine any challenge the students had throughout the course, because the course covers intense learning aspects (Lean UX and Scrum), in addition to a course project where students apply those techniques. Thus, "challenge" was included as a seed code.

The next step involved reading all the material and marking where the codes fit to the contents. During this process, some seed codes were reformulated, and others were split into more sub-codes when necessary. For example, the code "challenge" was decomposed into sub-codes to reflect the types of challenges reported about the integration of Lean UX into Scrum such as collaboration challenge, time challenge, and so on. New codes also emerged during the analysis process that represent interesting findings that can help us to understand other aspects. For example, the code "engagement" emerged from reading the students comments about the gamified part of the course to help us understand their perception of the benefits or drawback of including gamification. Formulating new codes and changing existing codes was one part of the analysis where the process was iterative because after that the authors had to go back to re-read and re-code the material again.

After coding the entire dataset, texts of similar and different codes were closely observed and compared accordingly, looking for patterns and themes. For example, we found the challenge of defining a testable and tactical hypothesis was mostly reported during the first sprint, which gave us the first clue about the steep learning curve issue discussed in section 4.2. The final step of the analysis process involved generating the lessons learned from the conclusions drawn in the previous step.

4.1 Integrating UX into Agile can be difficult to manage when working in short sprints

One of the most frequent challenges that we observed during the course and that was constantly reported by our students is the difficulty of fitting the Lean UX process into a short, one-week, sprint. This is a key issue in the case of our students, as there are no separate UX and development teams. One team designs, develops, and validates features in one-week iterations.

The students found that doing development and UX work jointly in one-week iterations was overwhelming, primarily because they are expected to deliver a product increment by the end of the sprint. Lean UX authors envision a typical two-week sprint in their discussions of the integration of Lean UX into Scrum. This gives the team one week to validate ideas through discovery work or implement part of the work without having to deliver a working increment. In our case, however, we were not able to extend the sprint duration due to the course time limits. Almost all teams from both years (2019 and 2020) managed to apply the Lean UX process within the three one-week sprints. However, they had to put in additional effort and allocate extra time to managing the integration process. Some students mentioned that this was an issue, as they found that defining hypotheses and running the design studio took longer than expected. Similar challenges related to not having enough time to carry out UX techniques were also reported in the literature [5].

Josh Seiden, the co-author of the Lean UX book, offers a few guidelines on how to handle UX work accommodated within a sprint [34]. Agile teams need to essentially think of UX work as a continuous part of the process instead of focusing on the finishing line. What cannot be done in one sprint can be broken down into multiple chunks, such that each chunk informs the decisions the team need to make by the end of the sprint. Accordingly, students were reminded in the second round of the course to think of UX as a continuous process, just like development. Additionally, students were instructed to carry out the "just enough" UX work that can inform the team's decisions on a certain feature or functionality. Questions for the team to discuss include: What do we need to validate or learn? How can we do that efficiently with the least waste? What is the most appropriate UX technique to apply here? Nonetheless, students continued to regard the integration of Lean UX into their one-week sprints as difficult to manage.

In terms of the course structure, it might help to redesign the course delivery format to follow a mixed approach combining the flipped classroom with traditional delivery in the form of PBL. In 2020, we had to deliver the course online due to the COVID-19 pandemic, and we followed this flipped-classroom approach. Lectures were recorded and students were asked to watch the videos before class time, so that class time is used for practical exercises. Yet, the short iterations issue was not fully fixed. A longer course duration with a longer sprint length could also help address this issue [7]. The course could be designed to span two semesters, with the second semester fully dedicated for a hands-on development project. This might also give the students the opportunity to engage with real stakeholders as mentors or clients. However, this has not been an option at the Universidad Politécnica de Madrid so far.

Table 1. Lessons learned from our experience in integrating Lean UX into Scrum in an educational context.

Lesson Learned	Suggested Action – Applied in our Course	Further Recommendations
Integrating UX into Agile can be difficult to manage when working in short sprints.	Remind the students to think of UX as a continuous process [34]. Adopt a flipped-classroom approach so students can gain some practice before the project development.	Design longer courses spanning two semesters to allow for longer sprint length, with the second semester entirely given over to a hands-on development project [7].
Learning Agile, Lean UX, and the integration of the two processes all at once constitutes a steep learning curve.	Offer a tailored learning experience based on the current knowledge of the students in agile, UX, and the integration of both processes.	Adopt a fully project-based learning approach to encourage active participation of the students in the learning process [7].
Defining testable and tactical “hypotheses” is perceived as the most challenging aspect of Lean UX.	Place more emphasis on the concept of hypothesis (examples and practice). Provide the students with weekly feedback on the team’s hypotheses. Adopt a flipped-classroom approach to facilitate interactive group learning activities on the hypothesis concept.	Adopt a systematic process to define and validate hypotheses modeled on hypothesis-driven development [35].
The high collaboration level of the Lean UX process can be difficult to maintain, particularly among distributed teams.	Optimize the learning environment for the students to enable real-time collaborative work (e.g., shared workspace, allocated time during the lecture, collaborative tools).	Use gamification mechanisms individually for each team member, instead of group mechanisms.
Integrating Lean UX into Scrum is an abstract process.	Continuously remind the students that Lean UX and Scrum integration is not a rigid process, and there is always space for flexibility as long as the principles are respected.	Include industry-education engagement activities in the course curriculum (e.g., workshops, involvement of real product owners, hackathons, etc.) to foster students’ innovation and autonomy [36].
When designed properly, gamification can be effective in motivating students to learn and apply the integration of Lean UX into Agile.	Employ the gamification mechanics, rewards and achievements to engage and motivate the students in the process of integrating Lean UX into Scrum.	Measure the effectiveness of gamification elements and incorporate them gradually [23], [37].

4.2 Learning Agile, Lean UX, and the integration of the two processes all at once constitutes a steep learning curve

In the first two sprints, students reported that they felt confused and overwhelmed by so many new Agile and Lean UX concepts and practices. The confusion might be logical, as they are dealing with the integration of Agile and UX for the first time with no, or little, prior knowledge and experience (the pre-experience questionnaires confirmed that most students were novice Agile practitioners, whereas they were all new to Lean UX).

Nonetheless, since part of the course was carried out following a project-based approach, we noticed how well the students’ knowledge of Agile and UX integration improved sprint after sprint in the course project.

For example, by the third sprint, students defined better hypotheses that were tactical and testable, ran the design studio

efficiently, and conducted proper user experiments to validate their hypothesis. This suggests that, although, as expected, integrating Lean UX into Scrum might constitute a steep learning curve with a tough initial learning process, proficiency grows with increased experience. Indeed, we found that the students’ knowledge of Lean UX had improved the third sprint.

One thing that might help to make the learning process more effective is to offer a tailored learning experience based on students’ current knowledge. We ran a quick survey at the beginning of the course to measure students’ knowledge and experience of Agile and UX (and Lean UX in particular). We also made sure that we provided students with constant feedback three times a week, twice via email used by students to provide updates with regard to their daily standups and the current progress of the development process, and then via the sprint review at the end of the sprint. The feedback mostly provided reinforcement of the

principles of Agile and UX [13]. Our aim was to offer minimal involvement to encourage students' autonomy and creativity.

The literature on SE education provides few additional strategies that can apply to the context of agile and UX integration. PBL is the main approach used to teach SE trending topics [7]. The active participation of the students in the learning process promotes critical thinking, skills that help the students manage the integration of agile and UX. The benefits of PBL for teaching Agile UX may even be amplified by following a flipped-classroom approach where time is allocated for the students to work on practical exercises collaboratively.

4.3 Defining testable and tactical hypotheses is perceived as the most challenging aspect of Lean UX

In both courses, we found that all teams had difficulties defining and writing their project hypotheses, particularly in the first two sprints. Students themselves confirmed this issue in the post-experience questionnaires and retrospective class. The hypothesis is one of the core Lean UX concepts. It represents a tactical and testable statement to frame ideas that require validation and experimentation. While there is a template to help teams translate their ideas into hypotheses, students had trouble grasping the concept, particularly the business outcome part.

At first, students thought that this issue might be specific to their limited knowledge and experience with the concept of the hypothesis. This might be true, but the definition of a good hypothesis is not a straightforward activity. This is why the authors of Lean UX designed two canvases as tools to help teams compose and prioritize hypotheses: the Lean UX Canvas [31] and Hypothesis Prioritization Canvas [38], respectively. Both techniques were available for the students to facilitate hypothesis discussion and creation. In fact, a two-hour class was set aside for a hands-on workshop where students worked on composing hypotheses using the Lean UX Canvas. However, we observed that most issues were related to the students' understanding of the hypothesis as a concept and its main components, namely, business outcome, user benefit, and the feature.

We noticed that students having difficulties defining a hypothesis with a business outcome of an appropriate relation or level of detail, like the associated user benefit and feature. In the initial sprints, most teams defined hypotheses that consist of a very high-level business outcome with a very specific functionality, where the functionality itself does not represent a whole feature and may not help the team to achieve the defined business outcome. As we observed during the sprint reviews, the confusion mostly revolved around the difference between a feature and a functionality. Because agile teams are used to thinking of features in terms of user stories that describe specific product functionalities, it was probably challenging to think more abstractly. For example, one team's project idea was similar to the idea of Kickstarter.com where entrepreneurs post business ideas

and ask venture capitalists to fund their businesses. This team defined a hypothesis as: *"We believe that we will increase our profit from selling entrepreneurial ideas if investors can invest in ideas using the list ideas feature"*. In this example, the "list ideas" feature is a simple functionality that provides a list of all the ideas posted by the entrepreneurs who need funding. In this case, the business outcome and the user benefit may not be achieved by this particular functionality. Instead, a compound feature such as "invest in ideas" or "fund an idea" may be more suitable. The "list ideas" functionality may form part of a bigger feature along with other functionalities.

The second issue was students defining a hypothesis with a feature that is irrelevant to the business outcome, that is, possibly does not directly contribute to achieving the business outcome. For example, another team's hypothesis states: *"We believe that we can increase the profit margin by 5% if busy users can save time when registering on our website using the one-click signup feature using their Google accounts"*. In this example, "one-click signup" (the feature) may benefit the business in general terms, but it may not directly contribute to "increasing the profit margin" (the business outcome).

The third and final issue was defining a hypothesis with a measurable business outcome. Most teams defined hypotheses with business goals that either cannot be easily measured or were poorly expressed resulting in an unclear real business value. For example, *"we believe that we will increase market awareness of our product if entrepreneurs can promote their business ideas using the share to social media feature"*. In this example, without quantifying or qualifying the level of market awareness by which the specified feature helps to increase it, it would not be possible to determine whether or not the business outcome is achieved.

Allocating more time and resources to explaining the hypothesis tactic may be effective in addressing this challenge. This can be done either during the lectures by providing more specific examples of hypotheses from different contexts or by introducing additional exercises on constructing hypotheses. In the 2020 course, we tried to focus more on explaining the hypothesis tactic through the lectures and provided more examples of different contexts. We also adopted a flipped classroom approach where we provided the students with videos explaining the hypothesis concept and then spent a whole class practicing hypothesis construction and validation using the Lean UX Canvas.

We also gave students weekly feedback on their sprint hypothesis. However, hypothesis creation was again an issue in the second round of the course. We continued to search for additional approaches to provide our students with a more systematic and structured approach to form tactical and testable hypotheses. Hypothesis-driven development [35] offers few tips on how to form testable and tactical hypotheses, such as establishing the metrics to be measured and success factor for benchmarking during hypothesis construction.

4.4 A high level of collaboration can be difficult to maintain throughout the Lean UX process, particularly within distributed teams

Our students reported difficulties in collaborating on every UX tactic of the process. The Lean UX process encourages the collaboration of the entire Agile team right from defining hypotheses through to running user experiments. This full-team involvement might be challenging in situations where team members have different working hours/schedules or are distributed across different locations. We observed that most collaboration issues among our students were caused by not being able to arrange face-to-face meetings on a regular basis. The issue of collaboration became even more evident in the 2020 course, when students could not manage to synchronize their meetings and activities due to COVID-19. In 2020, we had 14 students enrolled in the course from different time zones. Students also reported that, although collaboration during the design process was evidently beneficial, it was a challenging activity, as it is better conducted in face-to-face meetings with instant inputs from all team members.

Going back to the principles of Lean UX, it prioritizes collaboration and co-creation to ensure transparency and shared understanding among all team members. The philosophy should remain the same for remote teams or students with different schedules [39]. Educators should aim to optimize the learning environment for the students to enable real-time collaborative work by providing shared workspaces or collaborative tools.

External motivation techniques such as gamification can be helpful in keeping the team engaged in collaborative activities. We rewarded our students with additional points and badges (different from the course marks) for carrying out Lean UX activities collaboratively. This worked well with the class of 2019 when we received positive feedback from the students about the effectiveness of the team-based rewards. However, we received mixed feedback on the same technique in 2020. Students suggested that the points and badges should be given individually to each team member. This way, the active, as opposed to passive, participation of each team member is rewarded.

4.5 Integrating Lean UX into Scrum is an abstract process

Our students frequently complained that the Lean UX process is rather abstract, meaning that it lacks specific details on how to manage some activities when integrated into agile. This abstraction caused some frustration and confusion on how to efficiently manage UX work within the sprint. On one hand, all teams from both years stated that they found the process whereby Lean UX tasks are managed in the product backlog confusing. While the Lean UX process recommendations provided some level of detail, they also left room for each team to manage details on task management as per team preferences and objectives. The recommendations suggest either including Lean UX activities as

separate product backlog items (PBIs) within the same product backlog together with development work or incorporating each feature's development and UX work combined in one PBI. On the other hand, we noticed in the weekly user experiment tactic that the students did not understand which UX technique should be applied to validate the hypothesis or examine the value of the current feature. Indeed, students sometimes run an experiment whose goal is unclear, or it is not related to the hypothesis to validate.

Moreover, we left it to the students to decide whether to opt for a structured process to run the design studio formally or informally [30]. Some students preferred running the design studio informally because they found themselves communicating and collaborating more efficiently, while others preferred the formal structure as a way to engage the entire team. It may be necessary to continuously remind the students that there are different ways to apply Lean UX within Scrum. We very often had to encourage and help students to improvise a little when they could not figure out how to handle an integration challenge. Then, the students discussed how they managed such challenges and how they improved for the next sprint with the class during the sprint retrospective. Again, it may be useful to provide students with additional resources on UX to foster better decision making on how to run user experiments and collaborate efficiently in managing the design studio. We also think that including industry-education engagement activities can foster the students' innovation and autonomy [36]. In our course, we invited a guest speaker for a special seminar to share her experience in adopting Agile and how her team accounts for usability and UX aspects within the development process. In the future, we plan to host workshops with the collaboration of Agile practitioners and consultants.

4.6 When designed properly, gamification can be effective in motivating students to learn Agile UX

Although we received positive feedback on the use of gamification elements to facilitate the integration process, it is important to note that this may not always be the case. Based on observing the students during sprint reviews and analyzing projects reports, we confirmed that designing an impactful gamification strategy is not straightforward. From the feedback received, we found that rewards, particularly points and badges, have the most effective influence on students' engagement and motivation in applying Lean UX in Scrum (see Figure 3). According to the GLUX gamification strategy, badges were earned by the teams, for example, upon carrying out a Lean UX tactic for the first time or achieving a Lean UX challenge. The teams then displayed all the badges on a team achievements board.

This gave the students a more engaging experience integrating Lean UX with Scrum as badges provided visual representation of their team's achievements, while points motivated the students to apply Lean UX tactics more frequently.

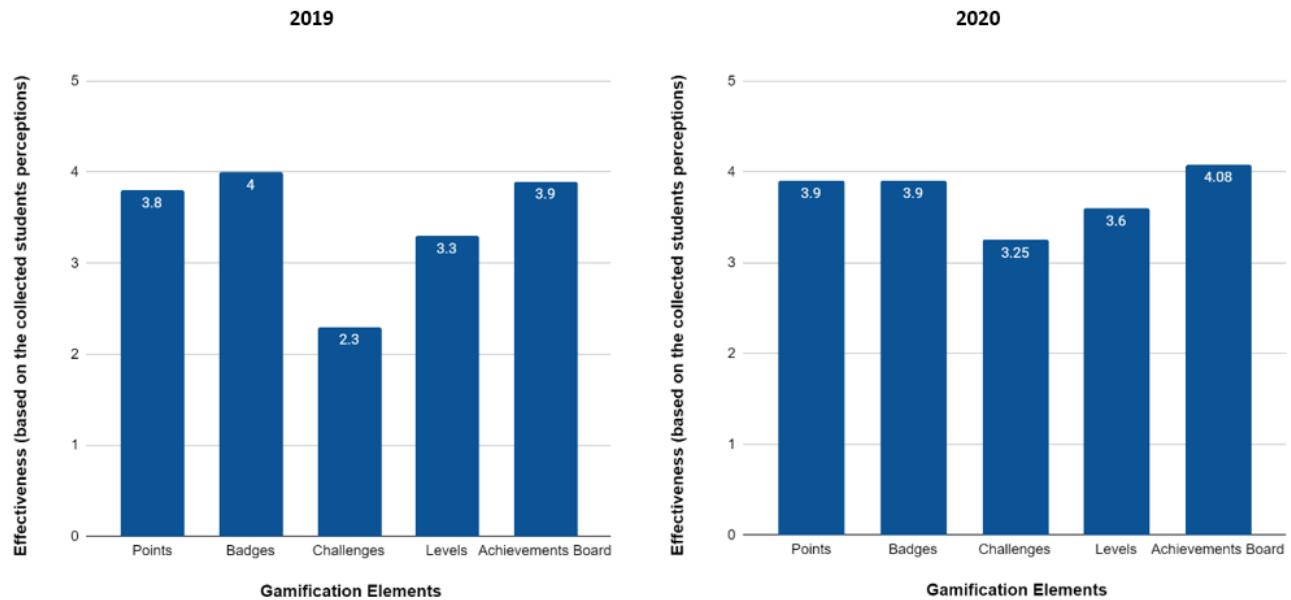


Figure 3. Students' rating of the effectiveness of each gamification element in improving their engagement and motivation in the Scrum and Lean UX integration process.

As shown in Figure 3, achievements is another gamification element that was found to be effective at engaging the students in integrating Lean UX with Scrum. As mentioned, GLUX offers a complementary technique called the achievements board, a virtual space where the team posts the points and badges that they earned during the development process. Students enjoyed this technique because it provided them with a dynamic and interactive visual space where they can track and share their awards and provided constant feedback on their progress in applying Lean UX. We also paid particular attention to the extent to which gamification helped the students to improve team collaboration. Our students thought that rewards were helpful in getting the whole team to collaborate.

However, not all the gamification elements were perceived with the same extent of effectiveness in making the process of Scrum and Lean UX integration an engaging process. For example, few students thought that the challenges presented in the GLUX guide added a pressure on the team to apply all Lean UX tactics in all sprints.

Additionally, as mentioned in Section 4.4 above, we noticed a difference in the feedback that we received from the 2019 class and the 2020 class on the reward mechanism. Students of the 2019 class enjoyed the team-based rewards system (where a point or a badge is earned by the team as whole). The students highlighted the benefit of team-based rewards, especially during the design studio tactic, where each and every team member must participate in the design session for the team to earn the associated rewards. On the other hand, students from the 2020 class thought that the rewards should be given individually for them to be recognized and appreciated. The 2020 class think that individual rewards would

introduce competition among team members and thus make the process more engaging and fun.

We should note, however, that gamification can be susceptible to the “novelty effect”, that is, the excitement and engagement with respect to applying Lean UX tactics may fade away after the first few iterations [37, 40]. This may not represent an issue in university education contexts where a course typically lasts one academic semester. However, it may be necessary to continuously monitor the outcomes of the gamified experience for prolonged experiences.

Either way, gamification should be used as a temporary solution that can help boost the player’s motivation towards adopting new habits and learning new skills. The temporary use of gamification is in fact recommended to avoid situations where users end up considering the whole process as a game and focus on collecting rewards and competing against each other, neglecting the primary goal of the original process [40, 41].

Besides, we recommend starting small to avoid wasting time and effort in designing a gamified process that may not work [23, 36]. The gamification elements should be incorporated into the process gradually, starting with one technique, say rewards. Once the applicability of the incorporated gamification element has been validated, additional gamification elements can be used to enforce the same or different objectives.

5 Conclusion

The integration of Agile and UX has received significant attention over the past years [3, 5]. Most of the research efforts in this field aimed at examining the different ways of integrating Agile and UX and the challenges of such integration in the software industry.

Meanwhile, the gap between Agile and UX in academia remains significant. In this paper, we reported our experience of designing and applying a novel course that covers the integration of Lean UX into Scrum for graduate SE students. Our aim is to mend the rift between Agile and UX in academia and foster a user-centered development approach. The course follows a hybrid approach of traditional lectures and exercises combined with PBL and employs gamification techniques to improve students' engagement. We briefly explained the structure of the course and how we used a recently developed framework, called GLUX, to guide the students in learning and applying the integration of Lean UX and Scrum. We identified six lessons learned that point to important aspects to consider when integrating Lean UX and Scrum in industrial and educational contexts. We also discuss how gamifying the process of integrating Lean UX into Scrum was well received by our students, as they particularly enjoyed receiving rewards and showing off their achievements.

This paper reports only one experience, and further empirical research is needed to validate the findings and critically evaluate GLUX and, in general, the different methods and techniques proposed in the literature for integrating UX processes and practices into Agile. However, we think that this paper can provide valuable insights for SE academics who are interested in Agile UX courses. Additionally, most of the reported insights are also potentially applicable in the industrial context, as some situations, like software practitioners not having a deep background in UX practices, are also common in this domain.

REFERENCES

- [1] O. Sohaib and K. Khan, "Integrating usability engineering and agile software development: A literature review," *2010 Int. Conf. Comput. Des. Appl. ICCDA 2010*, vol. 2, no. Iccda, pp. V2-32-V2-38, 2010.
- [2] T. S. Da Silva, A. Martin, F. Maurer, and M. Silveira, "User-centered design and agile methods: A systematic review," *Proc. - 2011 Agil. Conf. Agil. 2011*, pp. 77–86, 2011.
- [3] T. S. Da Silva, M. S. Silveira, F. Maurer, and F. F. Silveira, "The evolution of agile UXD," *Inf. Softw. Technol.*, vol. 102, no. March, pp. 1–5, 2018.
- [4] Digital.ai, "14th annual STATE OF AGILE REPORT," *Annu. Rep. STATE Agil.*, vol. 14, no. 14, pp. 2–19, 2020.
- [5] K. Curcio, R. Santana, S. Reinehr, and A. Malucci, "Usability in agile software development: A tertiary study," *Comput. Stand. Interfaces*, vol. 64, no. December 2018, pp. 61–77, 2019.
- [6] C. Péaire, "Dual-track agile in software engineering education," *Proc. - 2019 IEEE/ACM 41st Int. Conf. Softw. Eng. Softw. Eng. Educ. Training, ICSE-SEET 2019*, pp. 38–49, 2019.
- [7] O. Cico, L. Jaccheri, A. Nguyen-Duc, and H. Zhang, "Exploring the intersection between software industry and Software Engineering education - A systematic mapping of Software Engineering Trends," *J. Syst. Softw.*, vol. 172, p. 110736, 2021.
- [8] P. Ralph, "Improving coverage of design in information systems education," *Proc. 2012 Int. Conf. Inf. Syst.*, no. December 2011, pp. 1–15, 2012.
- [9] A. Clear and A. Parrish, *Computing Curricula 2020 - Paradigms for Global Computing Education*, 2020.
- [10] M. M. Alhammad, "A gamified framework to integrate user experience into agile software development process," ETSI_Informatica, 2020.
- [11] A. Ananjeva, J. S. Persson, and A. Bruun, "Integrating UX work with agile development through user stories: An action research study in a small software company," *J. Syst. Softw.*, vol. 170, p. 110785, 2020.
- [12] P. Kashfi, R. Feldt, and A. Nilsson, "Integrating UX principles and practices into software development organizations: A case study of influencing events," *J. Syst. Softw.*, vol. 154, pp. 37–58, 2019.
- [13] M. Brhel, H. Meth, A. Maedche, and K. Werder, "Exploring principles of user-centered agile software development: A literature review," *Inf. Softw. Technol.*, vol. 61, pp. 163–181, 2015.
- [14] G. Jurca, T. D. Hellmann, and F. Maurer, "Integrating agile and user-centered design: A systematic mapping and review of evaluation and validation studies of agile-UX," *Proc. - 2014 Agil. Conf. Agil. 2014*, pp. 24–32, 2014.
- [15] A. Y. Wale-Kolade, "Integrating usability work into a large inter-organisational agile development project: Tactics developed by usability designers," *J. Syst. Softw.*, vol. 100, pp. 54–66, 2015.
- [16] A. Bruun and J. Stage, "New approaches to usability evaluation in software development: Barefoot and crowdsourcing," *J. Syst. Softw.*, vol. 105, pp. 40–53, 2015.
- [17] C. Felker, R. Slanova, and J. Davis, "Integrating UX with scrum in an undergraduate software development project," *SIGCSE '12 - Proc. 43rd ACM Tech. Symp. Comput. Sci. Educ.*, pp. 301–306, 2012.
- [18] S. Deterding, D. Dixon, R. Khaled, and L. Nacke, "From game design elements to gamefulness," in *Proceedings of the 15th International Academic MindTrek Conference on Envisioning Future Media Environments - MindTrek '11*, 2011, p. 9.
- [19] O. Pedreira, F. Garcia, N. Brisaboa, and M. Piattini, "Gamification in software engineering – A systematic mapping," *Inf. Softw. Technol.*, vol. 57, pp. 157–168, 2015.
- [20] G. A. García-Mireles and M. E. Morales-Trujillo, "Gamification in Software Engineering: A Tertiary Study," *Adv. Intell. Syst. Comput.*, vol. 1071, pp. 116–128, 2020.
- [21] R. Cursino, D. Ferreira, M. Lencastre, R. Fagundes, and J. Pimentel, "Gamification in requirements engineering: A systematic review," *Proc. - 2018 Int. Conf. Qual. Inf. Commun. Technol. QUATIC 2018*, pp. 119–125, 2018.
- [22] L. Machuca-Villegas and G. P. Gasca-Hurtado, "Gamification for improving software project: Systematic mapping in project management," *Iber. Conf. Inf. Syst. Technol. Cist.*, vol. 2018-June, pp. 1–6, 2018.
- [23] M. M. Alhammad and A. M. Moreno, "Gamification in software engineering education: A systematic mapping," *J. Syst. Softw.*, vol. 141, pp. 131–150, 2018.
- [24] D. de P. Porto, G. M. de Jesus, F. C. Ferrari, and S. C. P. F. Fabbri, "Initiatives and challenges of using gamification in software engineering: A Systematic Mapping," *J. Syst. Softw.*, vol. 173, p. 110870, 2021.
- [25] M. M. Alhammad and A. M. Moreno, "What is going on in agile gamification?," *ACM Int. Conf. Proceeding Ser.*, vol. Part F1477, 2018.
- [26] M. M. Alhammad and A. M. Moreno, "Challenges of gamification in software process improvement," *J. Softw. Evol. Process*, no. February 2019, pp. 1–13, 2020.
- [27] M. Brhel, H. Meth, A. Maedche, and K. Werder, "Exploring principles of user-centered agile software development: A literature review," *Inf. Softw. Technol.*, vol. 61, pp. 163–181, 2015.
- [28] J. Nielsen, *Usability engineering*. Morgan Kaufmann, 1994.
- [29] N. Juristo, A. Moreno, and M. Sanchez-Segura, "Guidelines for Eliciting Usability Functionalities," *IEEE Trans. Softw. Eng.*, vol. 33, no. 11, pp. 744–758, 2007.
- [30] J. Gothelf and J. Seiden, *Lean UX: designing great products with agile teams*. "O'Reilly Media, Inc.", 2016.
- [31] J. Gothelf, "Lean UX Canvas V2," 2019. [Online]. Available: <https://jeffgothelf.com/blog/leanuxcanvas-v2/>.
- [32] P. Runeson, M. Host, A. Rainer, and R. Bjorn, *Case Study Research in Software Engineering - Guidelines an Examples*. John Wiley & Sons, 2012.
- [33] R. J. Wieringa, *Design science methodology: For information systems and software engineering*, 2014.
- [34] J. Seiden, "Here's what to do when user research doesn't fit in a sprint," 2019. [Online]. Available: <https://uxdesign.cc/heres-what-to-do-when-user-research-doesn-t-fit-in-a-sprint-2fb85db7d48c>.
- [35] A. Cowan, "HYPOTHESIS-DRIVEN DEVELOPMENT (PRACTITIONER'S GUIDE)," [Online]. Available: <https://www.alexandercowan.com/hypothesis-driven-development-practitioners-guide/>.
- [36] S. C. Paiva and D. B. F. Carvalho, "Software creation workshop: a capstone course for business-oriented software engineering teaching," in *Proceedings of the XXXII Brazilian Symposium on Software Engineering*, 2018, pp. 280–288.
- [37] K. Werbach and D. Hunter, "For the Win - Wharton School Press," *Wharton School Press*, 2015. [Online]. Available: <https://wsp.wharton.upenn.edu/book/for-the-win/>. [Accessed: 05-Aug-2020].
- [38] J. Gothelf, "The Hypothesis Prioritization Canvas," 2019. [Online]. Available: <https://jeffgothelf.com/blog/the-hypothesis-prioritization-canvas/>.
- [39] J. Follett, "What is Lean UX? Streamlining user experience for an increasingly agile world," 2017. [Online]. Available:

- [40] <https://www.oreilly.com/radar/what-is-lean-ux/>.
E. L. Deci, R. Koestner, and R. M. Ryan, “A meta-analytic review of experiments examining the effects of extrinsic rewards on intrinsic motivation.,” *Psychol. Bull.*, vol. 125, no. 6, p. 627, 1999.
- [41] I. Kuo, “Foursquare’s Removal of Gamification: Not a Mistake but a Mature Design Decision,” 2013. [Online]. Available: <https://www.gamification.co/2013/03/15/the-removal-of-foursquare-gamification/>.
- [42] T. Dal Sasso, A. Mocci, M. Lanza, and E. Mastrodicasa, “How to gamify software engineering,” *SANER 2017 - 24th IEEE Int. Conf. Softw. Anal. Evol. Reengineering*, pp. 261–271, 2017.



Approaches to manage the user experience process in Agile software development: A systematic literature review

Andreas Hinderks ^{a,*}, Francisco José Domínguez Mayo ^a, Jörg Thomaschewski ^b,
María José Escalona ^a

^a University of Seville, Spain

^b University of Applied Science Emden/Leer, Germany



ARTICLE INFO

Keywords:

User experience management
UX process
User experience
UX
Usability
HCI
Agile methods
Agile
Systematic literature review

ABSTRACT

Context: Software development companies use Agile methods to develop their products or services efficiently and in a goal-oriented way. But this alone is not enough to satisfy user demands today. It is much more important nowadays that a product or service should offer a great user experience — the user wants to have some positive user experience while interacting with the product or service.

Objective: An essential requirement is the integration of user experience methods in Agile software development. Based on this, the development of positive user experience must be managed. We understand management in general as a combination of a goal, a strategy, and resources. When applied to UX, user experience management consists of a UX goal, a UX strategy, and UX resources.

Method: We have conducted a systematic literature review (SLR) to analyse suitable approaches for managing user experience in the context of Agile software development.

Results: We have identified 49 relevant studies in this regard. After analysing the studies in detail, we have identified different primary approaches that can be deemed suitable for UX management. Additionally, we have identified several UX methods that are used in combination with the primary approaches.

Conclusions: However, we could not identify any approaches that directly address UX management. There is also no general definition or common understanding of UX management. To successfully implement UX management, it is important to know what UX management actually is and how to measure or determine successful UX management.

1. Introduction

Today's users expect a high level of satisfaction while interacting with a product. They expect to be able to use the product without any major effort to finish their tasks in a quick and efficient manner. Moreover, for a product to succeed, it is important to consider hedonic interaction qualities — i.e. those that are not directly target-oriented [1]. In summary, the user wants to have a positive user experience while interacting with any product or service.

A well-known definition of user experience is given in ISO 9241-210 [2]. Here user experience is defined as 'a person's perceptions and responses that result from the use or anticipated use of a product, system or service'. Therefore, user experience is viewed as a holistic concept that includes all types of emotional, cognitive, or physical reactions concerning the concrete or even only the assumed usage of a product formed before, during, and after use.

A different interpretation defines user experience as a set of distinct quality criteria [1] that includes the classical usability criteria or pragmatic qualities, such as efficiency, controllability, or learnability, and non-goal directed or hedonic quality criteria [3] like stimulation, novelty, or aesthetics [4]. This definition has the advantage that it splits the general notion of user experience into a number of quality criteria, thereby describing the distinct and relatively well-defined aspects of user experience.

Software development companies use Agile methods to develop products or services more efficiently. Agile methods (e.g. Scrum [5], Kanban [6], or Extreme Programming (XP) [7]) reduce the time taken to develop a product available in the market [8]. The iterative approach to developing software minimizes the risk of developing software that is not in line with what is needed in the market [9]. By performing

* Corresponding author.

E-mail addresses: andreas.hinderks@iwt2.org (A. Hinderks), fjdominguez@us.es (F.J. Domínguez Mayo), joerg.thomaschewski@hs-emden-leer.de (J. Thomaschewski), mjescalona@us.es (M.J. Escalona).

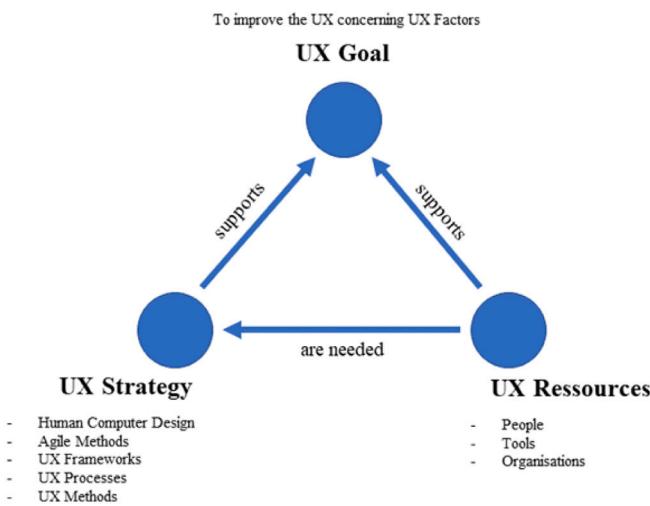


Fig. 1. User Experience Management based on McKeown [12].

retrospectives [5] at the end of an iteration, both product quality and Agile process quality can be improved.

To develop the best possible product with great user experience, it is essential to have the right management in place in terms of UX. To the best of our knowledge, there is no approved definition of UX management in the literature. There is also no common understanding of what UX management is or how to apply it (Section 2).

We generally understand management based on the explanations of Drucker [10] and Stone [11] — it is a combination of a goal, a strategy, and resources. When applied to UX, user experience management consists of a UX goal, a UX strategy, and UX resources (Fig. 1) based on the work of McKeown [12].

For example, a UX goal can be fixed to improve UX for a particular factor or quality criteria of UX. For this purpose, a UX strategy can be developed from different UX methods. For instance, to reach the UX goal, you can conduct a survey with a UX questionnaire, such as the User Experience Questionnaire [13] or the SUPR-Q [14], before and after the development. The UX strategy is that the results from the questionnaires after the development should be better than what they were before the development.

Both UX strategy and UX resources are necessary to achieve the UX goal. It should be known before the next development iteration, whose requirements positively supported the UX goal. In this way can the UX goal be achieved in a goal-oriented manner.

This paper reports the findings of a systematic literature review (SLR) in the field of approaches to manage the user experience process by focusing on Agile software development. This SLR will be addressed by the following research questions:

- RQ1: Which approaches are suitable for UX management in an agile context?
- RQ2: What conclusions can be deducted from the studies found?
- RQ3: How can user experience in Agile software development be planned and controlled for a product backlog item or a requirement before the development?
- RQ4: What retrospective proposals exist to improve the efficiency and effectiveness of the user experience process in terms of Agile software development?

This paper is structured as follows: Section 2 briefly summarizes the related work and presents gap analysis. Section 3 present the review method including research questions of this SLR, search strategy, selection process, quality assessment, and data extraction. Section 4 outlines the results and key findings of our study as well as the answers to our research questions. Section 5 discusses the meaning of the findings

and the limitations of our study. The paper ends with Section 6, with conclusions and ideas for future work.

2. Background and related work

As already mentioned in the introduction, we did not find a definition of UX management nor a common understanding of the term in the literature. We searched for “user experience management” and similar terms in IEEEXplore, Science Direct, Scopus, Springer Link and ACM. The full search string we used was: (“user experience management”) OR (“manage user experience”) OR (“ux management”) OR (“manage ux”). In the end, we found five relevant paper. In these paper, there are various approaches or descriptions of UX management. The term UX management is often used without any explanation. We present the five paper in the next two paragraphs.

In the literature, the term UX management is used differently. Szóstek [15] used the term UX management in the context of team building and empowerment. This includes career planning and development, team management, and training of individual team members. Anderson et al. [16] used a similar approach — in addition to building a UX team, they proposed that a C-level executive focus on user experience is necessary for UX management to have any corporate influence at all. For the implementation of UX management, Anderson et al. [16] and Rosenberg [17], for example, offered various patterns that provide support at the levels of planning, decision, tactics, and conflict.

Another approach is the use of a UX maturity model. The advantage of using such a model is that it determines the current maturity level of an organization. Thus, its weaknesses can be identified. But the decisive factor is which dimensions are mapped in the UX maturity model. For example, the *Total User Experience Management (TUXM)* [18] model contains elements such as UX objectives, integrated design system, strategic communication, continual improvement, fact-based decision-making, and a T-type design team. The *Nielsen Corporate Usability Maturity Model* [19], on the other hand, comprises dimensions such as the developers’ attitude towards usability, the management’s attitude towards usability, the usability practitioner’s role, usability methods and techniques, and strategic usability. At first glance, it is noticeable that the TUXM model contains the dimension called UX objectives which is not present in Nielsen’s model. Conversely, the Nielsen model is more focused on practical implementation. The testing of a suitable UX maturity model should be carried out before deployment and tailored to the needs of the organization [19].

To the best of our knowledge, we did not find any paper in the literature that considers both managing UX process and Agile software development. But we found papers that analyse the integration of UX or similar methods and Agile software development. Therefore, the next section summarizes an overview of SLRs regarding the integration of UX and Agile software development.

2.1. Summary of related literature reviews

In the literature, there are many reviews that investigate the integration of HCI and Agile methods. The term ‘Agile methods’ is used in the same way by all SLRs. However, there are differences in the processes or methods from the HCI area being used or integrated. The range of methods includes classic usability engineering, user-centred design (UCD) or human-centred design (HCD) [2], and UX methods in general, as well as design thinking.

The next paragraphs briefly summarize the SLRs found on the basis of our search results (Section 4.1). In Fig. 2, the SLRs are arranged in chronological order on a timeline.

The 2010 SLR by Bruun [20] investigated whether developers are trained in usability engineering so that they can apply usability engineering methods themselves. One of the main results of the SLR is the following finding: usability engineering is mainly published with

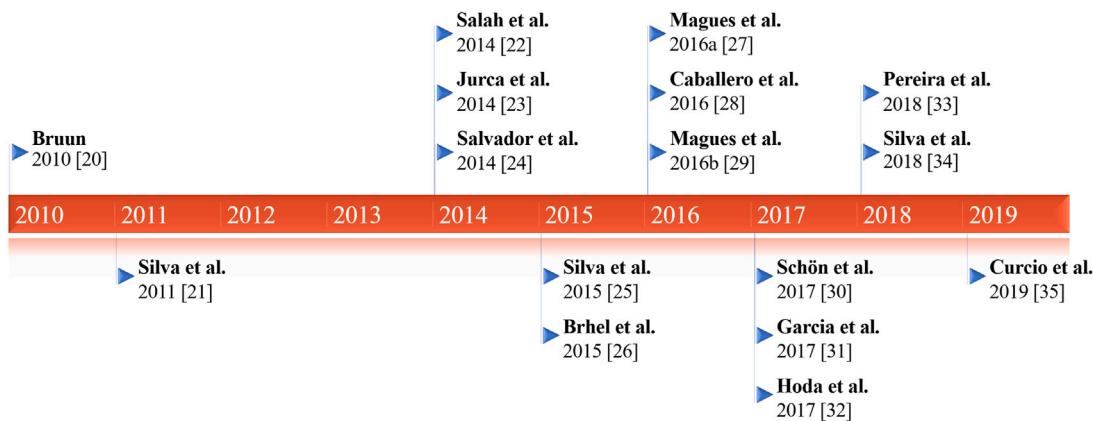


Fig. 2. Systematic Reviews in a Chronological Order.

a university or laboratory focus. Only a small part was dedicated to training the developers of usability methods. In the end, only one study could be identified that covered the essential aspects, namely user-based methods, training costs, focus on organizational contexts, and practitioners.

In 2011, Silva et al. [21] conducted an SLR on the integration of Agile methods and user-centred design. The authors analysed how usability problems are handled in Agile projects. The authors identified the following key aspects that play an essential role in integration: little upfront design, prototyping, user stories, user testing, inspection evaluation, and one sprint ahead.

In 2014, Salah et al. [22] analysed the current state of Agile and User-Centred Design Integration (AUCDI). Their analysis should identify the factors for the integration of Agile method and user-centred design. Besides, the authors examined the challenges and key aspects to ensure a successful integration. The identified key aspects are lack of allocated time for upfront activities, difficulty of modularization, optimizing the work between developers and UCD practitioners, performing usability testing, and lack of documentation.

In 2014, the SLR of Jurca et al. [23] analysed the literature to derive findings and recommendations for the integration of Agile and UX (Agile-UX). One finding is that Agile-UX methods are not anchored in companies and therefore do not receive the necessary support. Besides, UX designers are reworked and not part of the development team, but they are responsible for several development teams. This has been shown to reduce the efficiency and effectiveness of UX designers.

Salvador et al. [24] conducted in 2014 an SLR with the focus on which usability methods are used in Agile methods and when. The most commonly used usability methods include: fast prototyping, individual inquiry, formal tests, and heuristic evaluations. These methods are used about 50% during the implementation phase and 40% during the design phase. Only 10% of the usability methods used are implemented automatically.

In 2015, Silva et al. [25] performed a systematic mapping and analysed publications from significant Agile and HCI conferences. The objective was to answer the research questions on how Agile UCD is understood and which techniques are used in Agile UCD. Agile UCD is generally understood in the same way. It applies equally to the phases research, design, prototype, and evaluate. The most common technique is the implementation of usability test on lightweight prototypes.

In 2015, Brhel et al. [26] published an SLR by stating the principles of user-centred Agile software development (UCASD). The authors aimed to assess the current state of the art regarding the integration of Agile software development and user-centred design. Using a coding system, the authors extracted five derived principles: separate product discovery and product creation, iterative and incremental design and development, parallel interwoven creation tracks, continuous stakeholder involvement, and artefact-mediated communication.

In 2016, Magües et al. [27] conducted a systematic mapping study (SMS) in order to determine the current status of the integration of usability techniques in Agile processes. To that end, 31 studies were analysed and the usability techniques used were assigned to the development phase (requirements engineering, design, and evaluation). The most frequently used usability techniques for requirements engineering are ‘personas’; for design, ‘low-fi prototyping’; and for evaluation, ‘usability expert evaluations’.

Likewise in 2016, Caballero et al. [28] conducted a literature review to investigate the extent to which Agile teams integrated UCD methods in their Agile software development process. One result was that the most frequently used Agile methods are Scrum and XP. The three main UCD methods, which represent 70% of the methods used, are prototypes, user stories, and usability testing.

Also in 2016, the results from the SLR by Magües et al. [29] were further analysed in a mapping study by Magües et al. [27]. According to Brhel et al. [26], the selected studies were classified into the categories ‘process integration’ (48%), ‘practice integration’ (19%), ‘team integration’ (17%), and ‘technology’ (4%). The remaining papers could not be directly allocated. In conclusion, the authors concluded that there are no formalized suggestions for integrating usability techniques in Agile software development.

In 2017, the systematic literature review by Schön et al. [30] focused on approaches and methods for involving stakeholders in the process of Agile requirements engineering. A total of 27 papers were analysed. The most important result: there is no common understanding of the user perspective in Agile software development. However, four methods (Human-Centred Design, Design Thinking, Contextual Inquiry, and participatory design) were identified that integrate knowledge of user needs in Agile software development.

Garcia et al. [31] conducted a systematic mapping study in 2017. The purpose was to investigate artefacts used in communication between Agile methods and User-Centred Design. A total of 20 artefacts were identified and examined, such as prototype, user story, scenario, sketch, persona, and card like the design card or the task-case card. During the development iteration, about 56% of the artefacts are used. The rest are used during the discovery or planning phase.

In a meta-study in 2017, Hoda et al. [32] examined SLRs that treat Agile software development. A restriction to HCI was not made. The aim was to identify which developments in Agile software development can be recognized by the SLRs investigated. One finding is that the significant integration of established domains such as usability, CMMI, and global software engineering can be recognized. Usability is the second-most common (18%) integrated domain.

To evaluate how the Design Thinking approach is used in conjunction with Agile software development methods, Pereira & Russo [33] in 2018 used a systematic literature review. In total, 29 articles were

collected, categorized, and reviewed. The results show that most integrated models are applied throughout the software lifecycle. In most cases, the design thinking approach of the International Organization for Standardization (ISO) was integrated in Scrum as an Agile method.

In 2018, Silva et al. [34] analysed the results obtained by Brhel's SLR [26] concerning the state-of-the-art integration of Agile methods and the user experience design. The outcome from the respective publications was divided into three dimensions: process and practice, people and social, and technology and artefacts. As a result, the individual outcomes were arranged on a timeline so that the chronological sequence of the publications was visualized. The authors stated in their analysis that solutions are already being offered for the dimensions process and practice and people and social concerning integration. Finally, the authors concluded that technology and artefacts are still missing to integrate Agile methods and user experience design with Agile UXD.

In a meta-study conducted in 2019, Curcio et al. [35] examined SLRs concerning Agile methods and usability. The fundamental question concerned how usability methods could be integrated in Agile software development. It was found that there are different levels of integration — process, practices, team, and technology integration. The biggest challenges are issues related to tests, time, work balance, modularization, feedback, prioritization, and documentation. Another important finding is that the type of integration that has evolved from two independent teams (parallel track) to one team during the search period.

In total, we presented 16 SLRs for the integration of UX in Agile software development. The number of SLRs indicates that the integration has met the scientific interest. Besides, the SLRs show that everyone has a different focus on integration. Finally, the SLRs presented here show positive progress in the integration of UX in Agile software development.

2.2. Gap analysis

In a further step, we investigated the research questions of SLRs. We assigned each research question to the category UX strategy, UX resources, or UX goal depending on the objective of the research questions. The categorization was done based only on the purposes of the research questions. The results of the research questions were not further investigated. A total of 47 research questions from the 16 SLRs were examined. Twenty-nine research questions were assigned to the category UX strategy, 7 UX resources, and 0 UX goal. The remaining 11 research questions could not be assigned to any of the categories.

The results of the research show a focus on UX strategy. This is remarkable in that a UX strategy should always start with a UX goal as a prerequisite or objective. Only if a UX goal has been defined, a corresponding UX strategy can be selected. Every UX strategy indeed leads to a UX goal, but the definition of this goal is undefined and therefore, not manageable. From our point of view, all three categories have to be covered if managing UX is needed.

The studies from the related literature review deal in different levels with the integration of UX or HCI in Agile software development. Garcia et al. [31] and Bruun [20] for example provide approaches for measuring the success of the applied approaches in terms of improving UX. However, we found that the next step is to focus on the strategic pursuit of user experience improvement. It was assumed that the integration of UX methods in Agile software development improves the user experience of the product, but this cannot be measured. However, to determine whether a goal has been achieved, the previous, expected, and post-implementation state should be measured [12]. This is the only way to determine whether the UX strategy and UX resources used have achieved the UX goal (Fig. 3). In other words: Has desired UX been achieved?

We did not find a systematic literature review which investigates user experience management by focusing on Agile software development directly. The research questions tend to focus on UX strategies and perhaps UX resources, but not on UX goals. For this purpose, we conducted this SLR.



Fig. 3. Desired Outcome vs. Results based on the Strategic Planning Cycle [12].

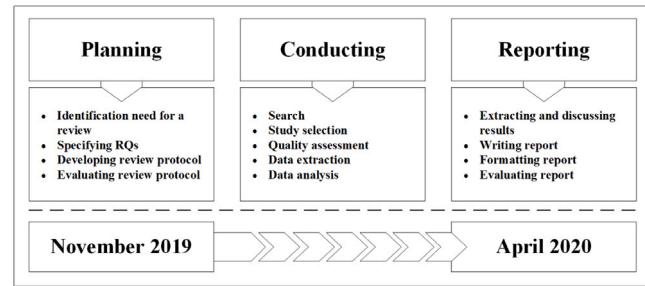


Fig. 4. Phases of an SLR.

3. Research methodology

Appropriate guidelines have been followed for conducting a systematic review, particularly the guidelines for SLR in software engineering by [36]. According to these guidelines, our SLR consists of three main phases. Fig. 4 shows the most important stages of each phase.

Owing to the high number of retrieved studies, we used the SLR Tool [37] and the software Citavi in order to manage information obtained in an efficient manner. We used the SLR tool when performing the SLR (managing the paper, developing the review protocol, documenting the search, and conducting quality assessment). In our literature database managed with Citavi, we imported the result of the SLR to use the management and citation functions.

3.1. Objectives and research questions

In the beginning, we did informal research on UX management or related terms. We conducted the informal research with Science Direct, Springer Link, IEEEXplore, Scopus, and ACM with the keyword ‘user experience management’ and variations of it.

The result was presented at the beginning of Section 2. However, during the research, we also found that the term ‘UX management’ is neither sufficiently defined nor explained in the literature. Further, we found through GAP analysis (Section 2.2) that there was a research gap in the goal, strategy, and resources concerning UX management. Besides, our informal research revealed that the number of papers found was too small and their content too widely scattered. However, we found approaches that allow UX management, as described in Section 2.2. This is the basis for our research questions.

RQ1: Which approaches are suitable for UX management in an agile context? This question aims to identify approaches that can potentially be used for UX management. We did not expect that the approaches that had been found could be used explicitly for UX management. Otherwise, we could have already identified approaches in the literature search (Section 2). Our analysis was intended to list approaches that were generally successful or had a high acceptance concerning UX methods in Agile software development.

Table 1
Keywords used for search.

Category	Keywords
Agile	Agile, Kanban, Scrum, Lean, Extreme Programming, Design Thinking
User Experience	User Experience, UX, Usability, HCD, HCI, HMI, UCD

RQ2: What conclusions can be deduced from the studies found?

Concerning these research questions, we wanted to find out whether, in addition to approaches, other findings on UX management could also be derived from the studies. Not every study contains an approach that can be used directly for UX management. We instead assumed that studies would be found which described the integration of user experience methods and Agile software development. These aspects also need to be considered and analysed.

RQ3: How can user experience in Agile software development be planned and controlled for a product-backlog item or a requirement before development? Management also implies a goal — what is to be achieved so that the necessary strategy and resources can be selected? The third research question aims to identify approaches that can be used to estimate UX product-backlog items before development. The result of the estimation is to figure out the potential UX that can be reached if the product-backlog item will be developed. In this way, the estimation of UX can be used to determine where there is potential to achieve a potential UX goal. Further, the question remains as to what extent the estimated UX can be expressed in the form of product-backlog items or requirements.

RQ4: What retrospective proposals exist to improve the efficiency and effectiveness of the user experience process in terms of Agile software development?

In terms of these research questions, our goal was to identify proposals for improving the UX process. We had to consider the fact that Agile software development was usually iterative. This means that after each iteration, there is the possibility of improving the UX process.

3.2. Search strategy and data sources

Based on the research questions and research objectives, we developed a search strategy. This strategy contains the search string, the search space, and the process to select the relevant papers.

The first step is to create a set of keywords. Since UX management has not been sufficiently covered in the literature (Section 2.2), the set of keywords consists of *Agile* and *user experience* as far as related terms are included. In practice, it has been shown that *Agile* is often not directly addressed, but rather *Kanban*, *Scrum* etc. Agile frameworks, like *Scrum*, or agile methods, like *Kanban*, are often used as a keyword in combination with Agile Methods. For this reason, we have included agile frameworks and methods in the search string. We also included the term *design thinking* because our experience has shown that useful publications have also been found with this term.

In the second step, we extended the keywords by alternative spellings and synonyms. These were extracted from the previously analysed literature (Section 2). Finally, we consolidated and optimized the list of keywords. The final list of keywords is shown in Table 1.

The set of keywords was then transferred to a search string in the next step. This is as follows:

(agile OR kanban OR scrum OR lean OR “extreme programming” OR “design thinking”)

AND

(“user experience” OR ux OR usability OR hcd OR hci OR hmi OR ucd)

Table 2
Search space with specification of search strategy (TAK = Title, Abstract, and Keywords) and number of paper.

Library	Search strategy	Number
IEEEExplore	Full Text	863
Science Direct	TAK	61
SCOPUS	TAK	1,308
SpringerLink	Full Text	3,874
ACM	TAK	26

This search string was adapted to the syntax of the respective search spaces as these had partially differed. The actual logic, however, had not been changed.

The search space included digital libraries, journals, and conference proceedings. A complete list of the search space is shown in Table 2. The search was conducted at all search spaces in January 2020.

Without any restriction – i.e. plain full-text search of the search engine – $N_{P0} = 44,637$ (Fig. 5) papers were found.

It should be noted that IEEEExplore and Springer Link had problems restricting the search to title, abstract, and keywords. Both did not offer the possibility to search for title, abstract, and keywords together. The conversation with the support of the respective providers has also led to no result. These problems were partially resolved by the owner of the search engines, but they led us to a slightly different strategy. Wherever possible, we downloaded the paper and put the abstract and keywords into plain text. From 4496 paper, we were able to extract the abstract and keywords from 2733 paper. Finally, we conducted an own search limited to title, abstract, and keywords using the SLR Tool [37] on all $N_{P1} = 4496$ paper. The result was that we had to check $N_{P1} = 4496$ (Fig. 5) paper initially.

By conducting the internal search function of the SLR Tool [37], we could reduce the result by searching only on title, abstract, and keywords so that the amount of paper to be examined was $N_{P2} = 1253$ (Fig. 5). This data set was further examined by us, as described in Section 3.3.

3.3. Study selection

In the previous section, it was described how the number of papers was limited by the search criteria (Step N_{P2}). In a further step, we reduced the number of papers from $N_{P2} = 1253$ to $N_{P3} = 196$ by scanning the title. We only included those papers that were interesting and valuable for our SLR in terms of their title. The title should be recognized that the paper is mainly about ‘agile’ and ‘user experience’. If the title indicated that the paper only applied agile and UX methods, the paper was excluded. In the following step, we reduced the number of papers to $N_{P4} = 110$ by reading the abstract. We applied the same criteria we used one step ago. All the decisions are traceably logged by the SLR tool.

In each step of the reduction, a set of selection criteria were applied. These which are divided into inclusion and exclusion criteria.

The inclusion criteria were: papers written in English; papers under peer-reviewed papers; and papers presenting approaches to integrate user experience methods (or similar) in Agile development processes.

Exclusion criteria were: no full books; papers whose full text were not available; papers only presenting lessons learned, ideas, guidelines or recommendations; papers introducing a panel talk or a workshop at a conference; papers with results that had already been published; papers that were not focused on Agile development; papers introducing tools whose underlying methodology was not comprehensibly described (black box).

After the study selection, we performed a snowballing process according to Wohlin [38]. We applied forward snowballing (search in papers that cite the paper) and backward snowballing (search in the reference list of the paper). Snowballing has the advantage that we

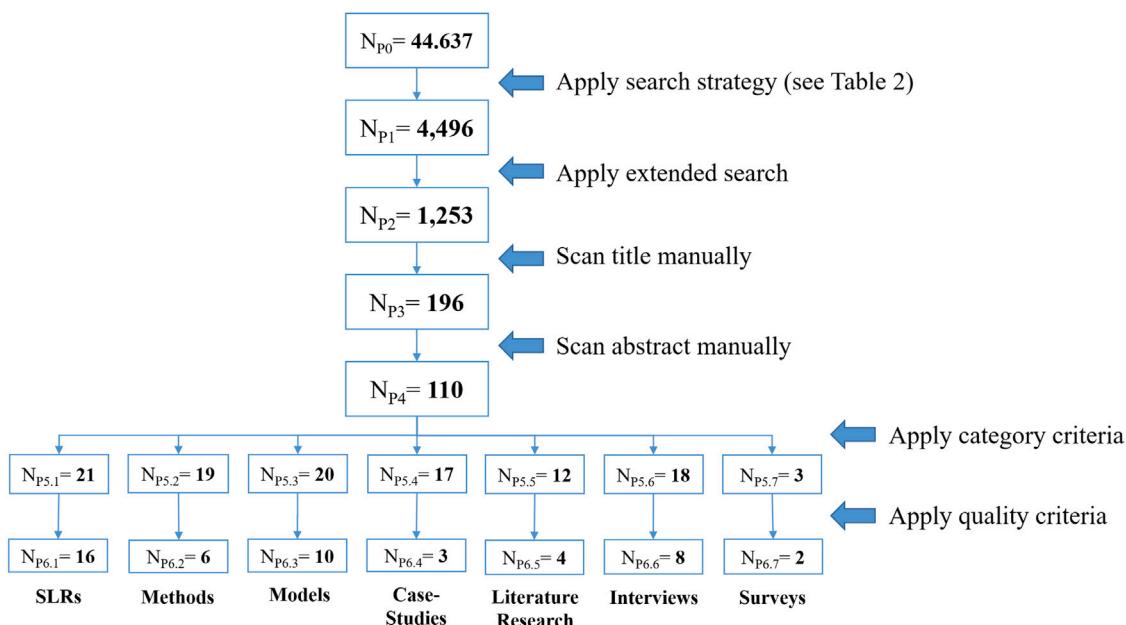


Fig. 5. Search Process comprising Phases and Inclusive Forward and Backward Snowballing.

Table 3
Quality checklist for empirical studies.

Item	Assessment criteria	Score	Description
QA1	Was more than one study conducted?	-1 0 1	Only one study was conducted Two studies were conducted More than two studies were conducted
QA2	Was the target group selected randomly?	-1 1	No, randomized group of participants Yes, randomized group of participants
QA3	Is the data analysis process appropriate?	-1 0 1	No analysis has been taken One statistical analysis has been taken The dataset is well analysed
QA4	Is the result of the statistical analysis appropriate?	-1 0 1	No or poor results The results are okay The results are good enough

can identify additional papers important to the SLR that were not identified via the SLR method itself in addition to the systematic search. The additionally found papers ($N = 29$) were inserted in step N_{p4} (Fig. 5). The numbers shown in Fig. 5 include the papers added by the snowballing. In total, seven additional papers for data extraction were added at the end.

3.4. Quality assessment

The papers selected in the previous section ($N_{p4} = 110$) were evaluated with a quality assessment. We developed a checklist (Table 3) based on the recommendations of Kitchenham and Charters [36] (Table 3) to evaluate case studies, literature research, interviews, and surveys. Methods and models were excluded because they are qualitative studies. To better classify the papers, we classified the papers according to case studies, literature research, interviews, surveys, methods and models. This classification is based on our own created system. We manually reviewed these methods and models by reading and evaluating the paper. The evaluation was based on the basic orientation of the study and whether it is suitable for our SLR.

The overall aim was to identify studies of low quality and then exclude them from our study.

In the end, every paper was rated with a sum of the individual result. We decided to include those articles with a score greater than or equal to 1.

The SLRs determined were checked to see whether the SLR was carried out in a traceable manner. Also, we checked whether the SLR was performed according to a standard published in the literature. The SLRs were then reduced to $N_{p6.1} = 16$.

Models and methods were generally reviewed for evaluation or validation. The aim was to determine whether the method or model was generally successfully applied in a study. After validation, we reduced the methods to $N_{p6.2} = 6$ and models to $N_{p6.3} = 10$.

After the user of the quality assessment, we reduced the case studies to $N_{p6.4} = 3$, literature research $N_{p6.5} = 4$, interviews $N_{p6.6} = 8$, and surveys to $N_{p6.7} = 2$.

In the end, we included 49 papers out of 110 papers in our SLR study.

3.5. Data extraction and analysis

According to Kitchenham's and Charters's guidelines [36], a form for data extraction was set up. We used the SLR Tool [37] in order to gather summaries, etc. The SLR Tool also supported the data extraction with regard to defined attributes from the protocol:

- Basic information: title, authors, publication date, DOI and URL
- Publication data: journal, conference, date (of conference), publisher, volume, issue, pages, keywords and abstract
- List of included references (if available)

Table 4
Distribution according to research methods and year.

Research method	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	Total	
SLR				1	1			3	2	3	3	2	1	16	32%
Methods					1			2		2			1	6	12%
Models	1	2		1	1	1		2		1	1			10	21%
Case studies						1			1	1			3		6%
Literature research							1	2			1		4		8%
Interviews	1	1						1	2	1	1		1	8	17%
Surveys							2						2		4%
Total	2	3	0	3	2	4	1	10	5	8	6	2	3	49	100%

In addition to the automatically extracted data, we have determined the following attributes manually:

- Paper category: e.g. SLRs, models, methods, case studies, literature research, interviews, and surveys
- Used UX methods: e.g. personas, prototypes, usability evaluations
- Agile methodology: e.g. Scrum, XP, Agile in general
- UX process integration: Parallel track or one track
- Development phase of usage: before, during, or after
- Short summary
- Results and contributions
- Personal assessment

In some cases, it was not possible to fill every attribute. In this case, we filled the attribute with ‘not specified’. In the last step, we checked the content of all studies and assigned them to particular research questions. The aim was to have a list of studies per research question so that we can use to answer them. The research questions are answered in the next section.

4. Results

In our work, we have selected 49 relevant studies. In this section, we will present the studies individually or in total, if necessary, to answer the research questions. The first part of this chapter gives an overview of the selected studies. In the second part, the individual research questions will be answered based on the studies.

4.1. Summary of the studies

Our search was limited to ‘Agile’ and ‘UX’ (and similar terms). An explicit restriction to UX management or similar was not made. This is because already in our first literature review it was found that the search result would not be sufficient to answer the research questions (Section 2). For this reason, the studies included in our SLR cover a broad spectrum. Incidentally, all the studies have been examined for their applicability in the field of UX management.

In journals, 15 (31%) of the included studies and 34 (69%) in conference proceedings were published. In Table 4, all 49 studies analysed in this SLR are grouped by research method and year. 15 (31%) of the studies were published between 2007 and 2013 and 34 (69%) between 2014 and 2019. It should also be noted that our research was conducted in January 2020, so there may still be late publications that are not included in this SLR.

All the 49 included studies were assigned different underlying research methods (Table 4). Of the studies included, 16 (32%) are *structured literature reviews* (SLR). These SLRs have not been used to answer the research questions as they themselves answer research questions that differ from ours. However, they have been presented in section Related Work (Section 2.1) and form the basis of the gap analysis (Section 2.2).

Further, six (12%) of the studies included in this study can be assigned to the category *Methods* [39–44]. Methods are to be understood rather generally in this category. Usually, a procedure is described,

how a problem was solved. A total of eight different methods were presented in these studies. Of the eight methods, four known methods were newly combined (Tool for A/B test [44], Checklist for a possible maturity model [43], Personas [40], Nielsen’s heuristics [42]), three new methods were presented (Usability Goals Achievement Metric (UGAM) [39], Index of Integration (IoI) [39], Web business process refactoring (WBPR) [41]), and one method was supplemented (UserX Story [42]).

The 10 (21%) *Models* [45–54] included can be divided into four frameworks, three conceptual models, two processes, and one lifecycle. For the sake of simplicity, they have been assigned to the research method models, since in some cases even the author of the study has not specified the research method.

All three (6%) *Case Studies* [55–57], four (8%) *Literature Research* [19,58–60], eight (17%) *Interviews* [55,61–67], and two (4%) *Surveys* [68,69] have been conducted in economic enterprises in such a way that the results and conclusions from each study are very practical. The *Case Studies* can be summarized as ‘classic case studies’. As *Literature Research*, we have categorized studies that answer research questions based on the literature. In this case, the difference between an *SLR* and *Literature Research* depends on the used method to conduct study. The *Interviews* were partly conducted as semi-structured interviews. Finally, the included *Surveys* were conducted in the classical way with a self-developed questionnaire.

Of the total of 49 studies, 16 *SLRs* are included, which we discussed in Section 2. Of the 33 remaining studies, 38 individual studies are included and analysed in this section. Two are *Multi-Case Studies* with four and two *Case Studies*, respectively, and two *Methods* are presented in one study. For this reason, 38 individual studies are selected and presented in our paper. The corresponding studies with several individual studies are numbered accordingly in round brackets. These are described in the next five sections. We further analyse the approaches in the 38 individual studies. We use the term approach as a generic term for a method, a model, a case study, literature research, an interview, or a survey.

4.1.1. User experience or usability

The 38 individual studies were examined as to whether each of them used the concept of *usability* or *user experience*. In some cases, it was not possible to assign the concept, as both concepts were used. In this case, the concept that was the most present in the individual study had been selected. Of the 38 individual studies, 16 (42%) use the concept *usability* and 22 (58%) the concept of *user experience*.

4.1.2. Used Agile method

We also examined the 38 individual studies according to the Agile method used. A total of 21 (55%) of the studies have not been assigned to a specific method. This is because the authors of the paper do not address explicit Agile methods. This means that it could be used for any Agile methods. The rest is divided between Scrum 11 (29%) and Extreme Programming 6 (16%).

4.1.3. Type of integration of UX in Agile methods

We also examined the 38 individual study to determine whether UX methods were used within the development team or outside as an additional parallel track/iteration. Of the individual studies examined, 11 (29%) integrated UX methods in the development team. The UX methods are usually used by UX professionals as well as developers. The eight (21%) individual studies, on the other hand, which use a UX team with its organization, in addition to the development team, are different. With an additional UX team, this team does not necessarily have to work exclusively with the development team, but can also work for several teams. The remaining 19 (50%) individual studies are not further specified for integration. One reason for this is that the individual study condenses results in such a way that the type of integration can no longer be deduced from them. Another reason may be that individual UX professionals only work with the development team and therefore do not constitute a team in their own right.

4.1.4. Proposed approaches

In the next step, we extracted the proposed approaches from the 38 studies. In total, we were able to extract 18 unique primary (Table 5) approaches from 24 individual studies out of the 38 studies. We analysed these in a further step. The results are presented in Sections 4.1.5 and 4.2.1.

4.1.5. Time period of use

In a further step, all 18 approaches were analysed with their temporal applicability in development. The aim was to examine each approach about the phase of development in which this approach is to be or was used. The breakdown was made according to before, during, or after development. If an approach can be used in several phases of development, it was also assigned to those phases. In total, 21 (88%) approaches can be used before development, while 15 (63%) approaches can be used during development and 13 (54%) after development.

In the next paragraphs, we answer the individual research questions from Section 1 using the 38 individual studies and the 18 primary approaches extracted from the 38 individual studies. All studies, except for the SLRs determined, serve as a basis.

4.2. (RQ1) which approaches are suitable for UX management in an agile context?

The first research questions seek to answer which of the 18 primary approaches extracted from the studies are suitable for the management of UX. One criterion for determining when an approach is suitable is when the presented approach has been successfully applied within Agile software development. Thus, the approach presented in the study is suitable for UX management for the first time. The quality assessment (Section 3.4) already ensures that the approach has been sufficiently evaluated or validated.

Table 5 lists all the primary approaches that were mainly presented or used in the individual studies. This means that in addition to the primary approaches, additional UX methods have been used. The main focus for answering RQ1 is on the primary approaches. Additionally used UX methods can support the primary approach, but they would not bring the desired success if used in isolation. The additional UX methods (Table 6) that were used in combination with the primary approaches are described in Section 4.2.2.

4.2.1. Primary approaches presented in studies

In the following sections, we present the highlights found in the included studies. In Table 5, we list all 18 primary approaches presented in the studies (Section 4.1.4). These approaches are potentially suitable for managing UX processes because they integrate UX methods into agile software development.

The most frequently identified approach is '*Upfront UCD Design*'. A second UCD team will be added to the actual development team. This team works in a parallel track (Section 4.1.3), always one iteration ahead of the development iteration. In this parallel track, prototypes [55,61,70] are usually created in various forms. These prototypes are then handed over to the development team and developed in the next iteration of iterations. Silva et al. chose a similar approach [50] — their framework defines the tasks of the individual teams within the parallel track concerning the user experience.

The second-most frequently used method or approach is '*Communication/Collaboration*' — a simple but successful approach. The approach is implemented in various ways. Ferreira et al. [55], for example, chose four different approaches in their study. The UI designers and developers worked together and constantly exchanged in this study. The UI designer usually developed a UI prototype based on a specification or a user story. This was then discussed together with the developers and then implemented by the development team. The result was that the developers had a much better understanding of the goal of the specification or user story. Ferreira et al. [70] chose a similar approach in another study.

Øvad et al. [56] and Øvad and Larsen [71] followed a different approach. Both approaches aim to teach the developers selected usability and UX methods so that they can use them independently in their development work. The approaches include '*A/B Testing*' [56,71] as well as '*Focus Workshops*' [71] and '*Contextual Interviews*' [71].

The '*Cruiser Lifecycle*' developed by Memmel et al. [45] aimed at integrating the methods of human-computer interaction (HCI) in Agile software development. The lifecycle mainly consists of three phases. The result of the first phase (Initial Requirements Up-Front) is a collection of artefacts like use case diagrams, scenarios, and prototypes. These are then processed further in the second phase (Initial Conceptual Phase). In the second phase, the release plan, system metaphor, UI design, and UI specification are developed. In the third phase (Construction & Test Phase), the product is developed and then its usability is evaluated. If new requirements are to be implemented, work can begin again from Phase 1. Xiong and Wang [48] and Humayoun et al. developed similar phase models [49]. All of them are based on the fact that there is a certain number of phases in which certain HCI or UX tasks are to be completed.

Singh [47] extended Scrum to '*U-Scrum*' by adding another role called '*Usability Product Owner*'. This is to ensure that the usability represented by the newly created role is already considered when requirements are created. Both the Product Owner (PO) and the Usability Product Owner (UPO) work on the same level, but with different focuses. The PO corresponds to the defined role in the Scrum Method [5]. In contrast, the UPO has to define a user experience vision. This should be considered in the requirements analysis. Finally, the UPO is responsible for the usability design and for creating the requirements together with the PO.

Wolkerstorfer et al. [46] integrated UCD methods within Extreme Programming (XP) to an '*Agile Usability Process*'. Before the first iteration of development, conducting user studies, personas, and usability tests are necessary. During the development, usability expert evaluations of the product increment are performed simultaneously with the unit tests established in XP. These provide direct feedback to the development as well as work for the next iteration.

Joshi et al. [39] created the '*Usability Goals Achievement Metric (UGAM)*'. We will present this approach in Section 4.4 in more detail. The second metric presented in this paper is '*Index of Integration (IoI)*' [39]. This metric represents, on a scale from 0 to 100, the level of integration of HCI activities in software development activities. With both UGAM and IoI, the development team can know how good their developed product regarding UX happens to be.

'*Web business process refactoring*' (WBPR), developed by Distante et al. [41], is a framework to capture the usability improvements of business process web applications. A WBPR includes the following

Table 5
18 Primary approaches presented in studies.

No	Primary approach	Studies	Total
1	Upfront UCD design	[55](3), [55](4), [61], [70](1)	4
2	Communication/Collaboration	[55](1), [55](2), [70](2)	3
3	Teaching UX Methods	[56], [71]	2
4	Cruiser Lifecycle	[45]	1
5	U-Scrum	[47]	1
6	Agile Usability Process	[46]	1
7	Inter-Combined Model	[48]	1
8	Usability Goals Achievement Metric (UGAM)	[39]	1
9	Index of Integration (IoT)	[39]	1
10	Three-Fold Integration Framework	[49]	1
11	UXD & AD Framework	[50]	1
12	Agile Usability Model	[52]	1
13	Agile UX Model	[51]	1
14	Web Business Process Refactoring (WBPR)	[41]	1
15	UserX Story	[42]	1
16	Checklist for User-Centeredness of Agile Processes	[43]	1
17	CSWR Framework	[44]	1
18	SIBAP	[53]	1
Total			24

Table 6
Methods used in combination with primary approaches.

No	Method	Studies	Total
1	Prototyping (Low/High)	[45], [61], [48], [63], [55](2), [55](3), [55](4)	7
2	Personas	[61], [46], [48], [40], [42], [54]	6
3	Task/Usage Scenarios	[45], [48], [54]	3
4	Acceptance Test	[55](2), [55](3)	2
5	Focus Groups	[45], [71]	2
6	Expert Reviews	[45], [46]	2
7	UX Questionnaires	[45], [48]	2
8	Contextual inquiry	[61], [71]	2
9	Usability Testing	[61], [63]	2
10	User Evaluation	[46], [54]	2
11	Interviews	[48], [54]	2
12	A/B Testing	[56], [71]	2
13	Card Sorting	[45]	1
14	Brainstorming	[45]	1
15	Usability Inspection	[45]	1
16	FlexREQ	[52]	1
Total			38

criteria: intent (usability qualities such as effectiveness, efficiency, and satisfaction), bad smell (an indicator of the lack of usability), motivation (description of the problem), and examples (describes the application). For each of these criteria, the authors provide instructions for creating WBPRs within the framework.

Navarro et al. [53] developed the '*Script-Based Aspect-Oriented GUI Prototyping (SIBAP)*' framework. The objective of the development was to create prototypes using a scripting language that can be used by both designers and developers. The prototype is the common artefact of designers and developers. The designers use the framework to create a prototype, which is then implemented by the developers. Thus, there is no media break between the prototyping of the designers and the development.

In summary, all the approaches presented in this section aim to promote a collaboration between UX professionals and developers. Some approaches directly promote communication, while others try to promote the exchange via a process or framework.

4.2.2. Additional UX methods included in primary approaches

UX methods are used in various ways. Usually, UX methods are used in addition to a process or a framework. Table 6 lists all the UX methods that were additionally used to the 24 primary approaches in the studies.

The two most frequently used methods are *Prototyping* and *Personas*. Prototyping and personas can be used as artefacts for the communication between UI designers and developers. The UI designers either develop a prototype together with the developers or work on it before the actual development. Personas, on the other hand, are usually

used permanently. Various methods are used to determine the requirements — these are *task/usage scenarios*, *focus groups*, *contextual inquiry*, *user evaluation*, *interviews*, *A/B testing*, *card sorting*, *brainstorming*, and *FlexREQ*. The following methods are used to measure and evaluate the user experience: acceptance test, expert reviews, UX questionnaires, usability testing, and usability inspection.

The UX methods listed in Table 6 were identified mainly by Jia et al. [68] as UX methods that serve in connection with the integration of UX in Agile software development. According to the list of Jia et al. [68] these are workshops, prototyping, interviews, scenarios, personas, field studies, usability goals, usability evaluation, questionnaires, and heuristic evaluation. What is interesting about the study, however, is the difference between the frequency of use and the evaluation of each individual method. The *Personas* method is rated quite positively, but 38% of the participants consulted it only once a year. In contrast, over 60% of the users use *Prototypes* at least two or three times a month.

4.3. (RQ2) what conclusions can be deducted from the studies found?

This question aims to identify findings that are useful in the context of UX management. To answer this research question, we analysed all 38 studies.

Both frameworks – user-centred design (UCD) and Agile methods – were developed independent of each other. Some studies have investigated to what extent the two frameworks can be integrated. The fundamental difference between the two frameworks is the focus. Agile methods focus on code production and the work of the developers,

whereas UCD focuses on the interface and the work of UX experts [58, 60, 66]. Agile methods do not support per se UCD methods, but they do not prevent them [60]. Communication between UX professionals and developers is one possible solution [60]. Additionally, a sprint zero can be performed to define the UX vision [66]. Another positive effect is that user research must be carried out before the actual development [66].

For both the UX professional and the developer, it is challenging to create a UX vision for a new product [62]. This is also confirmed by the study of Hokkanen et al. [64], who developed a list of UX qualities for new products. Besides, the study by Kuusinen [62] stated that too little time is spent on UX work and that individual disciplines could work better together.

There are several UX methods for being used within Agile software development (Section 4.2.2). Kuusinen [72] identified UX-related tasks in her study that can also be performed by a developer. These are tasks and not complete methods. For example, developers can clarify the user requirements. Developers can review UI designs or create their own UI designs. However, developers should be involved in the design decision so that the acceptance increases [72]. A retrospective view held by UX professionals and developers can improve the collaboration [72].

Various studies have also found that UX and Agile methods are not necessarily mutually exclusive [58, 60, 66]. Both UX and Agile methods are iterative; they support feedback; and they are multi-disciplinary [58]. However, how successfully UX methods are integrated and used in Agile methods considerably depends on the team itself [59].

The management does not consider UX to be part of the business strategy [69]. This means that while the management considers it essential to use UX methods to increase UX, it is not really part of the strategy [65]. One reason for this may be that UX knowledge is not yet firmly anchored in the management so as to become part of the company's strategy. UX knowledge is often expert knowledge and therefore not understandable or applicable for everyone [65].

4.4. (RQ3) how can user experience in agile software development be planned and controlled for a product backlog item or a requirement before development?

To answer the third research question, we examined all the approaches from Table 6 that can be used before development (Section 4.1.5). Two approaches [39, 42] were identified, but neither of them can meet the requirements of the research questions. However, since both the approaches are potentially suitable with limitations or extensions, we present them in the next sections.

Choma et al. [42] extended or supplemented the grammar of a user story with user experience aspects and usability requirements. New or replaced components of a *UserX Story* include personas, goals, interactions, contexts, and feedback. Nielsen's heuristics serve as the acceptance criteria. Expected user experience aspects can be specified as a heuristic. Based on these heuristics, the user experience could be estimated by extending and using a suitable method.

Joshi et al. [39] provide a *Usability Goals Achievement Metric (UGAM)*. This metric is calculated using individual parameters per usability quality (such as learnability, speed of use, and ease of use) weighted to a goal parameter score. This is the goal to be achieved.

After each usability evaluation, UX professionals calculate the achieved score based on the values from the usability evaluation. This makes it possible to determine whether the goal has been achieved by comparing the goal with the archived value. If the goal has not been achieved, it is possible to determine where it has not been achieved for each usability quality.

The two approaches are not directly based on product backlog items or requirements. Neither approach provides the possibility to estimate the user experience. In the end, both approaches can be used with an appropriate estimation method. Instead of the goal value, an estimated value of the user experience can be specified. The necessary prerequisites for a user experience value to be compared before development are given in both the approaches.

4.5. (RQ4) What retrospective proposals exist to improve the efficiency and effectiveness of the user experience process in terms of Agile software development?

To answer the fourth research question, we examined the included studies for approaches that improve the user experience process in its entirety, considering Agile software development. Thus, we identified two (6%) of the 33 studies that have been presented in the next two sections.

With the *Usability Goals Achievement Metric (UGAM)* developed by Joshi et al. [39], a goal for the user experience can be defined based on parameters or user experience aspects. By measuring after development, it can be determined to what extent the goal differs from actual reality. Further, the *Metric Index of Integration (IoI)* can be used to determine the maturity level of typical HCI activities in the development team. Both metrics (UGAM and IoI) can be used to perform a retrospective to identify the potential for improvement of the development of better user experience.

Nebe and Baloni [43] developed an Agile–HCD-Conformance Checklist based on the Checklist for User-Centredness of Agile Processes of DIN ISO 9241-210 [2]. Besides, best practices from Agile human-centred design approaches were included in the checklist. This checklist can serve as a source for the evaluation of the team's user-centeredness of Agile processes. In addition to recording the evaluation, the list also provides recommendations for improvement based on findings from the integration of UCD and Agile processes.

5. Discussion

This SLR allows us to identify and classify approaches to the applicability of UX management. In Section 4, we have provided answers to our research questions. In this section, we discuss the results, which have been divided as the following: a discussion of the results in general and specifically for each research question in Section 5.1. And Section 5.2 discusses the limitation of the results based on the method used.

5.1. Meaning of findings

In this SLR, we have examined 49 studies. As already mentioned in Section 3, we did not limit the search to UX management, but extended it to UX and Agile methods in general. As a result, the search result is very widespread. The individual studies mainly describe the integration of UX methods into Agile software development in partially different ways. All the studies have the common goal to develop a better product or service with a high UX through the integration. We have used this goal as the basis for the answers to the research questions. None of the studies addressed UX management directly.

The ratio of the number of SLRs investigated in the remaining studies is rather interesting. A total of 16 SLRs (32%) and 33 (68%) other studies were investigated. The integration of UX methods in Agile software development has been addressed and discussed in the literature. However, there is a noticeable lack of sufficiently validated approaches that support integration. In the selected studies, we were able to investigate approaches that support integration but do not explicitly address UX management. We discuss this in detail in the next sections.

5.1.1. Findings related to RQ1

The '*Upfront UCD Design*' was identified as the most frequently used approach. This approach is criticized in some cases [55, 60, 72] because it does not resolve the matter of integration of UX methods directly in Agile software development. It is only an approach that coordinates the work of two teams — the UX team and the development team. In this respect, integration can only be described as limited.

All studies have more or less one thing in common: they all support the cooperation and communication between the team members. This is either a direct component and goal of the approach [55,60,70,72] or is achieved indirectly through the methods used. It should be noted that the team members usually consist of two rather different groups — the UX team and the development team. Both use different methods to do their work. For example, the UX team uses the methods listed in Table 5.

However, it turns out that using a single UX method is not the solution. Rather, it shows that UX methods are always integrated into a superordinate structure (framework, process, lifecycle, etc.). The superordinate structure provides an expected result, whereas the UX methods support the result. So, both are necessary.

5.1.2. Findings related to RQ2

As in the discussion of the first research questions, the answer to the second research question shows that the cooperation between the two disciplines is the primary approach. However, the most essential two findings can be summarized as follows:

1. UX methods or HCI and Agile methods have been developed independent of each other. There are similarities, such as iterations and feedback, but they were not developed with the intention that they could be integrated. Nevertheless, both methods are not mutually exclusive.
2. In management, UX is not perceived as a business strategy. This inevitably means that decisions regarding the use of UX methods within Agile software development must be made at the team level. In terms of the Agile method, this is feasible, but it will never have the value of a company-wide business strategy.

Therefore, future research studies should focus more on the integration of UX and Agile as well as on management in equal measure.

5.1.3. Findings related to RQ3

Two methods were found to answer the third research questions. These methods, *UserX Story* [42] and *Usability Goals Achievement Metric (UGAM)* [39], were not developed for estimating the user experience for a product backlog item, but they are basically suitable. However, further development of both methods is necessary to enable them to represent the estimated UX.

In essence, no approaches were found to estimate the user experience for a product backlog item or requirement before development. However, UX management needs a way to estimate or capture the UX before development (Section 2.2). Only then can it be measured after development whether the desire outcome has been achieved (Section 2.2).

5.1.4. Findings related to RQ4

The answers to the fourth and last research questions resulted in two approaches: *Usability Goals Achievement Metric (UGAM)* [39] in combination with *Index of Integration (IoI)* [39] and the *Agile–HCD Conformance Checklist* [43]. However, both approaches aim to determine the current status of integration and the application of UX methods. Although this is suitable for determining weaknesses and thus pointing out possible improvement potential, this is not explicitly addressed.

On the other hand, in Scrum, as the primary representative of Agile methods, the Sprint Retrospective [5] is a primary component. This team event should allow the team to improve each member or the team. All this will help to develop it more efficiently and in a better way in the next iteration. In none of the studies investigated was the Sprint Retrospective used to improve the UX processes or the UX methods used.

5.2. Limitation of the review

We have used a predefined protocol for conducting this study to ensure its completeness. We may not have identified some relevant studies due to the large number of existing studies. We have minimized this risk by using forward and backward snowballing. Owing to the lack of digital recording and organization of the studies in some cases, a residual risk cannot be ruled out.

Another possible weakness of our approach might be the chosen search string. We searched for UX methods in combination with Agile software development. However, it is quite conceivable that some existing UX have not been published in combination with Agile software development, even though they are suitable for the stated purpose.

Regarding the limitations of data extraction, we are aware that some aspects may not have been sufficiently documented in the studies analysed. For this reason, our results might have been different if the studies had been documented more accurately. We have tried to address this problem by conducting a comprehensive quality assessment of the studies included.

6. Conclusions and future work

This paper presents an SLR about managing the UX process to identify suitable approaches for user experience management. The SLR was conducted according to the guideline offered by Kitchenham and Charters [36]. In an initial search, we found 44,637 studies. Our search process reduced the number of studies to 1253. We analysed these studies by their titles and abstracts and performed a quality assessment. These measures helped us to select 49 studies, including seven studies that were selected through a snowballing process.

This SLR has different implications for both practitioners and scientists. Based on the explanations in ‘Related Work’ (Section 2), we can summarize that approaches and methods are used to develop a better user experience, but not goal-oriented by defining a UX goal. Furthermore, there is no definition of UX management or a common understanding of UX management.

The approaches identified in the studies deal with the integration of UX methods or HCI in Agile methods. Upfront UCD design, communication/collaboration, and teaching UX methods have been applied in several studies. All other approaches have only been presented in the respective study without being found in other studies. In addition to the use of the approaches, several UX methods have been identified that have been used in combination with the approaches. The three most frequently identified UX methods are prototyping (low/high), personas, and task/usage scenarios. Among the approaches and UX methods analysed, it has been found that only one approach makes it possible to define a UX goal before development and to test it after development [39]. All other approaches aimed to systematically integrate the UX methods into Agile software development. Many of them can be used for UX management if they are adapted accordingly.

In conclusion, it can be summarized that UX management is neither sufficiently described nor used in a targeted manner. Further research may focus on a process model or lifecycle that would take account of a desired UX goal and figure out whether the goal is reached. To this end, we will develop a UX lifecycle that includes a UX goal, a strategy to reach the UX goal, methods to interpret the outcome, and a UX retrospective to improve the usage of the UX lifecycle.

CRediT authorship contribution statement

Andreas Hinderks: Conceptualization, Methodology, Validation, Formal analysis, Investigation, Data curation, Writing – original draft, Visualization, Project administration. **Francisco José Domínguez Mayo:** Methodology, Writing – review & editing, Supervision. **Jörg Thomaschewski:** Conceptualization, Validation, Writing – review & editing, Supervision. **María José Escalona:** Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

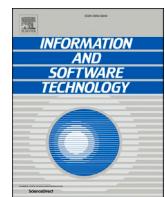
Acknowledgements

This research was partially supported by the POOLAS project (TIN2016-76956-C3-2-R) of the Spanish Government's Ministry of Economy and Competitiveness and by the NDT4.0 project (US-1251532) of the Junta de Andalucía, Spain.

References

- [1] J. Preece, Y. Rogers, H. Sharp, *Interaction Design: Beyond Human-Computer Interaction*, fourth ed., Wiley, Chichester, 2015.
- [2] ISO9241-210, *Ergonomics of human-system interaction - Part 210: Human-centred design for interactive systems*, 2020, ISO 9241-210:2020.
- [3] M. Hassenzahl, The effect of perceived hedonic quality on product appealingness, *Int. J. Hum.-Comput. Interact.* 16 (13(4)) (2001) 481–499.
- [4] N. Tractinsky, Aesthetics and apparent usability, in: S. Pemberton (Ed.), *The SIGCHI Conference*, 1997, pp. 115–122, <http://dx.doi.org/10.1145/258549.258626>.
- [5] K. Schwaber, *Agile Project Management with Scrum*, in: Microsoft professional, Microsoft Press, Redmond, Wash, 2004.
- [6] D.J. Anderson, *Kanban: Successful Evolutionary Change for Your Technology Business*, Blue Hole Press, Sequim, Washington, 2010.
- [7] K. Beck, C. Andres, *Extreme Programming Explained: Embrace Change*, second ed., 6. printing, in: *The XP series*, Addison-Wesley, Boston, 2007.
- [8] P. Serrador, J.K. Pinto, Does Agile work? — A quantitative analysis of agile project success, *Int. J. Project Manag.* 33 (5) (2015) 1040–1051, <http://dx.doi.org/10.1016/j.ijproman.2015.01.006>.
- [9] B. Boehm, R. Turner, Using risk to balance agile and plan- driven methods, *Computer* 36 (6) (2003) 57–66, <http://dx.doi.org/10.1109/MC.2003.1204376>.
- [10] P.F. Drucker, *The Practice of Management*, HarperCollins, New York, NY, 2009.
- [11] J. Magretta, N.D. Stone, *What Management Is: How It Works and Why It's Everyone's Business*, Profile Books, London, 2013.
- [12] M. McKeown, *The Strategy Book: How to Think and Act Strategically to Deliver Outstanding Results*, third ed., Pearson, Harlow, England, 2020.
- [13] M. Schrepp, J. Thomaschewski, *Handbook for the Modular Extension of the User Experience Questionnaire*, 2019.
- [14] J. Sauro, *SUPR-Q: A comprehensive measure of the quality of the website user experience*, *J. Usability Stud.* 2015 (10) (2015) 68–86.
- [15] A. Szóstek, A look into some practices behind microsoft UX management, in: J.A. Konstan, E.H. Chi, K. Höök (Eds.), *Proceedings of the 2012 ACM Annual Conference Extended Abstracts on Human Factors in Computing Systems Extended Abstracts - CHI EA '12*, ACM Press, New York, New York, USA, 2012, p. 605, <http://dx.doi.org/10.1145/2212776.2212833>.
- [16] R.I. Anderson, J. Ashley, T. Herrmann, J. Miller, J. Nieters, S.S. Eves, S.T. Watson, Moving ux into a position of corporate influence, in: M.B. Rosson, D. Gilmore (Eds.), *CHI '07 Extended Abstracts on Human Factors in Computing Systems - CHI '07*, ACM Press, New York, New York, USA, 2007, p. 1905, <http://dx.doi.org/10.1145/1240866.1240920>.
- [17] D. Rosenberg, The business of UX management, *Interactions* 26 (3) (2019) 28–35, <http://dx.doi.org/10.1145/3318131>.
- [18] H.B.-L. Duh, J.-J. Lee, P.L.P. Rau, M.Q. Chen, The management model development of user experience design in organization, in: P.-L.P. Rau (Ed.), *Cross-Cultural Design*, in: *Lecture Notes in Computer Science*, vol. 9741, Springer International Publishing, Cham, 2016, pp. 163–172, http://dx.doi.org/10.1007/978-3-319-40093-8_17.
- [19] D. Salah, R. Paige, P. Cairns, Integrating agile development processes and user centred design- A place for usability maturity models? in: S. Sauer, C. Bogdan, P. Forbrig, R. Bernhaupt, M. Winckler (Eds.), *Human-Centred Software Engineering*, in: *Lecture Notes in Computer Science*, vol. 8742, Springer Berlin Heidelberg, Berlin, Heidelberg, 2014, pp. 108–125, http://dx.doi.org/10.1007/978-3-662-44811-3_7.
- [20] A. Bruun, Training software developers in usability engineering, in: E. Hvannberg, M.K. Lárusdóttir, A. Blandford, J. Gulliksen (Eds.), *Proceedings of the 6th Nordic Conference on Human-Computer Interaction Extending Boundaries - NordiCHI '10*, ACM Press, New York, New York, USA, 2010, p. 82, <http://dx.doi.org/10.1145/1868914.1868928>.
- [21] T. Silva da Silva, A. Martin, F. Maurer, M. Silveira, User-centered design and agile methods: A systematic review, in: 2011 AGILE Conference, IEEE, 2011, pp. 77–86, <http://dx.doi.org/10.1109/AGILE.2011.24>.
- [22] D. Salah, R.F. Paige, P. Cairns, A systematic literature review for agile development processes and user centred design integration, in: M. Shepperd, T. Hall, I. Myrtevit (Eds.), *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering - EASE '14*, ACM Press, New York, New York, USA, 2014, pp. 1–10, <http://dx.doi.org/10.1145/2601248.2601276>.
- [23] G. Jurca, T.D. Hellmann, F. Maurer, Integrating agile and user-centered design: A systematic mapping and review of evaluation and validation studies of Agile-UX, in: 2014 Agile Conference, IEEE, 2014, pp. 24–32, <http://dx.doi.org/10.1109/AGILE.2014.17>.
- [24] C. Salvador, A. Nakasone, J.A. Pow-Sang, A systematic review of usability techniques in agile methodologies, in: Unknown (Ed.), *Proceedings of the 7th Euro American Conference on Telematics and Information Systems*, in: EATIS, vol. 14, ACM Press, New York, New York, USA, 2014, pp. 1–6, <http://dx.doi.org/10.1145/2590651.2590668>.
- [25] T. Silva da Silva, F.F. Silveira, M.S. Silveira, T. Hellmann, F. Maurer, A systematic mapping on agile UCD across the major agile and HCI conferences, in: O. Gervasi, B. Murgante, S. Misra, M.L. Gavrilova, A.M.A.C. Rocha, C. Torre, D. Taniar, B.O. Apduhan (Eds.), *Computational Science and Its Applications, ICCSA 2015*, in: *Lecture Notes in Computer Science*, vol. 9159, Springer International Publishing, Cham, 2015, pp. 86–100, http://dx.doi.org/10.1007/978-3-319-21413-9_7.
- [26] M. Brhel, H. Meth, A. Maedche, K. Werder, Exploring principles of user-centered agile software development: A literature review, *Inf. Softw. Technol.* 61 (2015) 163–181, <http://dx.doi.org/10.1016/j.infsof.2015.01.004>.
- [27] D.A. Magües, J.W. Castro, S.T. Acuña, Usability in agile development: A systematic mapping study, in: 2016 XLII Latin American Computing Conference, CLEI, IEEE, 2016, pp. 1–8, <http://dx.doi.org/10.1109/CLEI.2016.7833437>.
- [28] L. Caballero, A.M. Moreno, A. Seffah, How agile developers integrate user-centered design into their processes: A literature review, *Int. J. Softw. Eng. Knowl. Eng.* 26 (08) (2016) 1175–1201, <http://dx.doi.org/10.1142/S0218194016500418>.
- [29] D.A. Magües, J.W. Castro, S.T. Acuna, HCI usability techniques in agile development, in: 2016 IEEE International Conference on Automatica, ICA-ACCA, IEEE, 2016, pp. 1–7, <http://dx.doi.org/10.1109/ICA-ACCA.2016.7778513>.
- [30] E.-M. Schön, J. Thomaschewski, M.J. Escalona, Agile requirements engineering: A systematic literature review, *Comput. Stand. Interfaces* 49 (2017) 79–91, <http://dx.doi.org/10.1016/j.csi.2016.08.011>.
- [31] A. Garcia, T. Silva da Silva, M. Selbach Silveira, Artifacts for agile user-centered design: A systematic mapping, in: *Proceedings of the 50th Hawaii International Conference on System Sciences* (2017), in: *Proceedings of the Annual Hawaii International Conference on System Sciences*, Hawaii International Conference on System Sciences, 2017, <http://dx.doi.org/10.24251/HICSS.2017.706>.
- [32] R. Hoda, N. Salleh, J. Grundy, H.M. Tee, Systematic literature reviews in agile software development: A tertiary study, *Inf. Softw. Technol.* 85 (2017) 60–70, <http://dx.doi.org/10.1016/j.infsof.2017.01.007>.
- [33] J.C. Pereira, R.d.F. Russo, Design thinking integrated in agile software development: A systematic literature review, *Procedia Comput. Sci.* 138 (2018) 775–782, <http://dx.doi.org/10.1016/j.procs.2018.10.101>.
- [34] T.S. Da Silva, M.S. Silveira, F. Maurer, F.F. Silveira, The evolution of agile UXD, *Inf. Softw. Technol.* 102 (2018) 1–5, <http://dx.doi.org/10.1016/j.infsof.2018.04.008>.
- [35] K. Curcio, R. Santana, S. Reinehr, A. Malucelli, Usability in agile software development: A tertiary study, *Comput. Stand. Interfaces* 64 (2019) 61–77, <http://dx.doi.org/10.1016/j.csi.2018.12.003>.
- [36] B. Kitchenham, S. Charters, *Guidelines for performing systematic literature reviews in software engineering*, 2007.
- [37] A. Hinderks, M. Schrepp, F.J. Domínguez Mayo, M.J. Escalona, J. Thomaschewski, UEQ KPI Value Range based on the UEQ Benchmark, <http://dx.doi.org/10.13140/RG.2.2.34239.76967>.
- [38] C. Wohlin, Guidelines for snowballing in systematic literature studies and a replication in software engineering, in: M. Shepperd, T. Hall, I. Myrtevit (Eds.), *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering - EASE '14*, ACM Press, New York, New York, USA, 2014, pp. 1–10, <http://dx.doi.org/10.1145/2601248.2601268>.
- [39] A. Joshi, N.L. Sarda, S. Tripathi, Measuring effectiveness of HCI integration in software development processes, *J. Syst. Softw.* 83 (11) (2010) 2045–2058, <http://dx.doi.org/10.1016/j.jss.2010.03.078>.
- [40] L. Caballero, A.M. Moreno, A. Seffah, Persona as a tool to involving human in agile methods: Contributions from HCI and marketing, in: S. Sauer, C. Bogdan, P. Forbrig, R. Bernhaupt, M. Winckler (Eds.), *Human-Centred Software Engineering*, in: *Lecture Notes in Computer Science*, vol. 8742, Springer Berlin Heidelberg, Berlin, Heidelberg, 2014, pp. 283–290, http://dx.doi.org/10.1007/978-3-662-44811-3_20.
- [41] D. Distante, A. Garrido, J. Camelier-Carvajal, R. Giandini, G. Rossi, Business processes refactoring to improve usability in E-commerce applications, *Electron. Commerce Res.* 14 (4) (2014) 497–529, <http://dx.doi.org/10.1007/s10660-014-9149-0>.
- [42] J. Choma, L.A.M. Zaina, D. Bernaldo, UserX story: Incorporating UX aspects into user stories elaboration, in: M. Kurosu (Ed.), *Human-Computer Interaction. Theory, Design, Development and Practice*, in: *Lecture Notes in Computer Science*, vol. 9731, Springer International Publishing, Cham, 2016, pp. 131–140, http://dx.doi.org/10.1007/978-3-319-39510-4_13.

- [43] K. Nebe, S. Baloni, Agile human-centred design: A conformance checklist, in: S. Yamamoto (Ed.), Human Interface and the Management of Information: Information, Design and Interaction, in: Lecture Notes in Computer Science, vol. 9734, Springer International Publishing, Cham, 2016, pp. 442–453, http://dx.doi.org/10.1007/978-3-319-40349-6_42.
- [44] S. Firmenich, A. Garrido, J. Grigera, J.M. Rivero, G. Rossi, Usability improvement through A/B testing and refactoring, *Softw. Qual. J.* 27 (1) (2019) 203–240, <http://dx.doi.org/10.1007/s11219-018-9413-y>.
- [45] T. Memmel, F. Gundelsweiler, H. Reiterer, Agile human-centered software engineering, in: Proceedings of the 21st British HCI Group Annual Conference on People and Computers: HCI...But Not As We Know It - Vol. 1, in: BCS-HCI '07, BCS Learning & Development Ltd., Swindon, GBR, 2007, pp. 167–175.
- [46] P. Wolkerstorfer, M. Tscheligi, R. Sefelin, H. Milchrahm, Z. Hussain, M. Lechner, S. Shahzad, Probing an agile usability process, in: M. Czerwinski, A. Lund, D. Tan (Eds.), Proceeding of the Twenty-Sixth Annual CHI Conference Extended Abstracts on Human Factors in Computing Systems - CHI '08, ACM Press, New York, New York, USA, 2008, p. 2151, <http://dx.doi.org/10.1145/1358628.1358648>.
- [47] M. Singh, U-SCRUM: An agile methodology for promoting usability, in: Agile 2008 Conference, IEEE, 2008, pp. 555–560, <http://dx.doi.org/10.1109/Agile.2008.33>.
- [48] Y. Xiong, A. Wang, A new combined method for UCD and software development and case study, in: The 2nd International Conference on Information Science and Engineering, IEEE, 2010, pp. 1–4, <http://dx.doi.org/10.1109/ICISE.2010.5690032>.
- [49] S.R. Humayoun, Y. Dubinsky, T. Catarci, A three-fold integration framework to incorporate user-centered design into agile software development, in: M. Kurosu (Ed.), Human Centered Design, in: Lecture Notes in Computer Science, vol. 6776, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 55–64, http://dx.doi.org/10.1007/978-3-642-21753-1_7.
- [50] T. Silva da Silva, M. Selbach Silveira, F. Maurer, T. Hellmann, User experience design and agile development: From theory to practice, *J. Softw. Eng. Appl.* 05 (10) (2012) 743–751, <http://dx.doi.org/10.4236/jsea.2012.510087>.
- [51] A.L. Peres, T.S.D. Silva, F.S. Silva, F.F. Soares, C.R.M.D. Carvalho, S.R.D.L. Meira, Agileux model: Towards a reference model on integrating UX in developing software using agile methodologies, in: 2014 Agile Conference, IEEE, 2014, pp. 61–63, <http://dx.doi.org/10.1109/AGILE.2014.15>.
- [52] S.M. Butt, A. Onn, M.M. Butt, N.T. Inam, S.M. Butt, Incorporation of usability evaluation methods in agile software model, in: 17th IEEE International Multi Topic Conference 2014, IEEE, 2014, pp. 193–199, <http://dx.doi.org/10.1109/INMIC.2014.7097336>.
- [53] P.L.M. Navarro, G.M. Pérez, D.S. Ruiz, A script-based prototyping framework to boost agile-UX developments, *J. Comput. Sci. Tech.* 31 (6) (2016) 1246–1261, <http://dx.doi.org/10.1007/s11390-016-1695-6>.
- [54] M. Aguiar, C. Zapata, Integrating UCD and an agile methodology in the development of a mobile catalog of plants, in: M. Soares, C. Falcão, T.Z. Ahram (Eds.), Advances in Ergonomics Modeling, Usability & Special Populations, in: Advances in Intelligent Systems and Computing, vol. 486, Springer International Publishing, Cham, 2017, pp. 75–87, http://dx.doi.org/10.1007/978-3-319-41685-4_8.
- [55] J. Ferreira, J. Noble, R. Biddle, Agile development iterations and UI design, in: AGILE 2007, AGILE 2007, IEEE, 2007, pp. 50–58, <http://dx.doi.org/10.1109/AGILE.2007.8>.
- [56] T. Øvad, N. Bornoe, L.B. Larsen, J. Stage, Teaching software developers to perform UX tasks, in: B. Ploderer, M. Carter, M. Gibbs, W. Smith, F. Vetere (Eds.), Proceedings of the Annual Meeting of the Australian Special Interest Group for Computer Human Interaction on - OzCHI '15, ACM Press, New York, New York, USA, 2015, pp. 397–406, <http://dx.doi.org/10.1145/2838739.2838764>.
- [57] T. Øvad, L.B. Larsen, How to reduce the UX bottleneck – train your software developers, *Behav. Inf. Technol.* 35 (12) (2016) 1080–1090, <http://dx.doi.org/10.1080/0144929X.2016.1225818>.
- [58] L. Schwartz, Agile-user experience design: an agile and user-centered process? in: ICSEA 2013: The Eighth International Conference on Software Engineering Advances, 2013, pp. 346–351.
- [59] L. Schwartz, Agile-user experience design: Does the involvement of usability experts improve the software quality? in: *International Journal on Advances in Software*, Vol. 7, 2014.
- [60] M. Larusdottir, J. Gulliksen, A. Cajander, A license to kill – improving UCSD in agile development, *J. Syst. Softw.* 123 (2017) 214–222, <http://dx.doi.org/10.1016/j.jss.2016.01.024>.
- [61] D. Fox, J. Sillito, F. Maurer, Agile methods and user-centered design: How these two methodologies are being successfully integrated in industry, in: Agile 2008 Conference, IEEE, 2008, pp. 63–72, <http://dx.doi.org/10.1109/Agile.2008.78>.
- [62] K. Kuusinen, Improving UX work in scrum development: A three-year follow-up study in a company, in: S. Sauer, C. Bogdan, P. Forbrig, R. Bernhaupt, M. Winckler (Eds.), Human-Centered Software Engineering, in: Lecture Notes in Computer Science, vol. 8742, Springer Berlin Heidelberg, Berlin, Heidelberg, 2014, pp. 259–266, http://dx.doi.org/10.1007/978-3-662-44811-3_17.
- [63] T.S.D. Silva, M.S. Silveira, F. Maurer, Usability evaluation practices within agile development, in: 2015 48th Hawaii International Conference on System Sciences, IEEE, 2015, pp. 5133–5142, <http://dx.doi.org/10.1109/HICSS.2015.607>.
- [64] L. Hokkanen, K. Kuusinen, K. Väinänen, Minimum viable user experience: A framework for supporting product design in startups, in: H. Sharp, T. Hall (Eds.), Agile Processes, in Software Engineering, and Extreme Programming, in: Lecture Notes in Business Information Processing, vol. 251, Springer International Publishing, Cham, 2016, pp. 66–78, http://dx.doi.org/10.1007/978-3-319-33515-5_6.
- [65] P. Kashfi, A. Nilsson, R. Feldt, Integrating user experience practices into software development processes: implications of the UX characteristics, *PeerJ Comput. Sci.* 3 (2017) e130, <http://dx.doi.org/10.7717/peerj.cs.130>.
- [66] N. Pillay, J. Wing, Agile UX: Integrating good UX development practices in agile, in: 2019 Conference on Information Communications Technology and Society, ICTAS, IEEE, 2019, pp. 1–6, <http://dx.doi.org/10.1109/ICTAS.2019.8703607>.
- [67] M.K. Larusdottir, Usability evaluation in software development practice, in: D. Hutchison, T. Kanade, J. Kittler, J.M. Kleinberg, F. Mattern, J.C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M.Y. Vardi, G. Weikum, P. Campos, N. Graham, J. Jorge, N. Nunes, P. Palanque, M. Winckler (Eds.), Human-Computer Interaction – INTERACT 2011, in: Lecture Notes in Computer Science, vol. 6949, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 430–433, http://dx.doi.org/10.1007/978-3-642-23768-3_30.
- [68] Y. Jia, M.K. Larusdottir, A. Cajander, The usage of usability techniques in scrum projects, in: D. Hutchison, T. Kanade, J. Kittler, J.M. Kleinberg, F. Mattern, J.C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M.Y. Vardi, G. Weikum, M. Winckler, P. Forbrig, R. Bernhaupt (Eds.), Human-Centered Software Engineering, in: Lecture Notes in Computer Science, vol. 7623, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 331–341, http://dx.doi.org/10.1007/978-3-642-34347-6_25.
- [69] K. Kuusinen, K. Väinänen-Vainio-Mattila, How to make agile UX work more efficient, in: L. Malmborg, T. Pederson (Eds.), Proceedings of the 7th Nordic Conference on Human-Computer Interaction Making Sense Through Design - NordiCHI '12, ACM Press, New York, New York, USA, 2012, p. 139, <http://dx.doi.org/10.1145/2399016.2399037>.
- [70] J. Ferreira, H. Sharp, H. Robinson, Agile development and user experience design integration as an ongoing achievement in practice, in: 2012 Agile Conference, IEEE, 2012, pp. 11–20, <http://dx.doi.org/10.1109/Agile.2012.33>.
- [71] T. Øvad, L.B. Larsen, Templates: A key to success when training developers to perform UX tasks, in: G. Cockton, M. Lárusdóttir, P. Gregory, A.S. Cajander (Eds.), Integrating User-Centred Design in Agile Development, in: Human–Computer Interaction Series, Springer International Publishing, Cham, 2016, pp. 77–96, http://dx.doi.org/10.1007/978-3-319-32165-3_3.
- [72] K. Kuusinen, Task allocation between UX specialists and developers in agile software development projects, in: J. Abascal, S. Barbosa, M. Fetter, T. Gross, P. Palanque, M. Winckler (Eds.), Human-Computer Interaction – INTERACT 2015, in: Lecture Notes in Computer Science, vol. 9298, Springer International Publishing, Cham, 2015, pp. 27–44, http://dx.doi.org/10.1007/978-3-319-22698-9_3.



Agile software development and UX design: A case study of integration by mutual adjustment

John Stouby Persson ^a, Anders Bruun ^{a,*}, Marta Kristín Lárusdóttir ^b, Peter Axel Nielsen ^a

^a Department of Computer Science, Aalborg University, Denmark

^b Department of Computer Science, Reykjavík University, Iceland



ABSTRACT

Context: Agility is an overarching ideal for empirically-driven software development processes that embrace change in order to improve quality, economy, and simplicity. While the pursuit of Agility has held prominence in software practice and research for over two decades, user experience (UX) designers struggle to integrate their work processes with agile software development.

Objective: As empirical processes are constantly evolving, so is this integration struggle for UX designers. We, therefore, present an industrial case study of how a Danish software company integrates UX design and agile software development.

Method: We conducted a case study involving (a) one iteration of individual interviews with 10 employees (four UX designers, three software developers, two project managers, and one solution architect) and (b) a follow-up iteration consisting of a workshop with 6 employees (three UX designers, two solution architects, and one project manager) two years later. We analyzed how the company's approach to integration with 'upfront design' and 'work in parallel' involve mutual adjustments as opposed to assimilation or separation of UX design and software development.

Results: Our analysis shows how integration through mutual adjustments made distinct contributions to UX designers' and software developers' pursuit of Agility. They experienced notably different work processes that still dealt effectively with change and contributed to quality, economy, or simplicity. Nevertheless, as shown from a follow-up workshop two years after our first interviews, these processes were still susceptible to integration struggles over time.

Conclusion: We conclude that integration based on mutual adjustment potentially makes Agility for UX designers and software developers different and mutually complementary. This integration contrasts with assimilation, which potentially makes their Agility mutually indistinguishably, and with separation, which makes their Agility different and mutually competing.

1. Introduction

Making software systems easy to use has increasingly become a prioritized goal for software development teams over the last three decades. Additionally, giving the users good experiences before, during, and after using software systems has become more acknowledged. Professionals have specialized in this focus and have many different roles, like user experience (UX) designer, UX analyst, UX evaluator, and UX manager [1]. In this paper, we focus on the role of UX designers. To achieve a good user experience, UX designers struggle with influencing software developers; and software developers struggle to stay agile while collaborating with UX designers.

Software developers, for their part, have for more than two decades been influenced by the Agility of software processes. This concern has been explicit since the Agile Manifesto [2] appeared in 2001, but Larman and Basili [3] trace it back to much earlier. According to the state of agile survey, the most popular agile development process is Scrum, with over 80% of participants using that process or some deviations of the

process [4]. Other processes mentioned in the survey are Extreme Programming and Lean. Initially, agile development processes were rooted in the software development industry, but lately, agile methodologies are spreading across a broad range of industries [5,6]. However, the substantial literature on agile software development does not provide an unequivocal and standard meaning to the concept of agile processes.

UX designers, who specialize in interaction design and areas different from the particular programming and technical development, struggle to integrate their work into agile processes [7,8]. They may see themselves as "add-ons" to agile development, despite their importance to the success of software projects [9]. User-centered methods and techniques such as comprehensive field investigation and thorough user testing may stand in stark contrast to the quick releases of working code valued in agile development processes. The sometimes conflicting concerns of software developers and UX designers are challenging for integrating their individual efforts for the success of a shared project. To better understand how such integration of work processes is carried out in practice, we present a case study of this integration at a Danish

* Corresponding author.

E-mail address: bruun@cs.aau.dk (A. Bruun).

software company to answer the research question:

1.1. How can integrated UX design and software development processes maintain their agility?

With this case study, we report how integrating software development and UX design can rely on mutual adjustment with upfront design and work in parallel. Our analysis, using Conboy's taxonomy of Agility [10], shows the distinct contributions to the software developers' and UX designers' Agility from mutual adjustment. However, these contributions to Agility, revisited in a follow-up workshop in the software company two years later, can dwindle with increased separation. With these insights, our study contributes to the extant research on integrating UX design with software development based on a theoretical and empirical grounded analysis of Agility.

The following Section 2 presents a review of the related literature on agile software development, UX design, and the concepts of Agility and integration. Next, Section 3 presents our case study method, including our choice of the case set in a company with a strong profile in UX design and agile software development and our data collection from 10 individual interviews during the first iteration, a second iteration workshop with 6 participants, and qualitative content analysis. Our findings in Section 4 are on their *upfront design and work in parallel*, followed by insights from a workshop for *reflecting on integration*. We finally discuss in Section 5 how our findings contribute to research and practice, followed by a conclusion in Section 6.

2. Related literature

2.1. Agile software development and UX design

The literature on agile software development processes initially had little concern for research on organizational Agility [10,11] and UX design [12]. A theoretical comparison of Scrum and Kanban's fit for UX activities shows that Kanban offers more flexibility and therefore fits better for integrating UX activities into the process [13]. A recent literature review states that the challenges of integrating UX activities into Scrum are related to: the insufficient importance assigned to UX activities in general; insufficient communication between UX designers and developers; insufficient resources assigned to upfront activities in Scrum, and customers trying to represent final users without being aware of their real needs [14]. Another literature review states that the challenges of integrating UX in agile development include lack of time to perform upfront-design and tests with real users, power struggle between UX designers and developers, lack of vision for the whole UX project, difficulty to prioritize UX activities, and lack of documentation [15]. On the contrary, Da Silva et al. [16], state that currently, there is a complete understanding in both communities that UX activities need to be integrated into agile development, and these can not be two separate processes, but more work is still to be done on the tools that address the integration of the agile and UX activities.

One approach to integrating UX activities into agile development is to fit lightweight UX activities into Scrum by using lightweight and incremental processes, conducting simplified think-aloud with pair test users combined with heuristic evaluation [17]. The benefits gained by that integration included more satisfied users; close understanding of users and their needs; better collaboration with stakeholders; and improvements in the development process, including reduced rework. However, neither Scrum nor Kanban supports UX effectively, and they refer to customers significantly more often than to users. Moreover, many processes are tailored to the specifics of development companies, situations, and conditions [9]; for example, having a customer on site in the development room is neglected due to the customer's reluctance to commit the necessary effort or merely the absence of an identifiable customer.

Another approach to tailoring an agile process to the integration of

UX activities involves being one iteration ahead by designing the user interface needed for the next iteration [18]. Customer data are gathered up front before the coding starts, design is done one sprint ahead, and user testing is conducted in the next sprint of the coding of that functionality. A systematic review suggests to: (a) conduct little design up front, (b) focus on close collaboration between UX designers and development experts, and (c) make UX designers work one sprint ahead of developers [19]. However, balancing the amount of upfront work and synchronizing between UX designers and software developers is one of the main challenges for their integration [20]. The lack of focus on usability and UX activities in agile software development has attracted growing research interest [8,12,21]. UX designers may have a broad set of responsibilities in agile projects, including sales and business development, which rely on a customer focus rather than a user-centered focus [1]. The UX designers, among other things, struggle with making low-fidelity prototypes at the start of the project and gathering feedback on those, getting collaboration from developers for creating the design, having too short time within each iteration to exploit different design options, and too little time to conduct an evaluation with real users, [22]. Similarly, a lack of collaboration and communication between UX designers and developers was noted by Kuusinen et al. [23], and a lack of understanding of the business and customer needs. A case study documents how UX designers constantly need to justify UX activities and employ salesmanship at the same time to have customers pay for the UX activities [1]. A more recent approach is to integrate Lean UX into agile [24–26]. The Lean process complements the disciplines of agile and UX design with the approach of validated learning using the Lean UX cycle of build, measure, and learn. They suggest that user tests are conducted each week to learn from the feedback gathered from users [24]. Zorzetti et al. [26] conclude that adopting an approach of LeanUX can bring changes in mindsets, activities, practices, and techniques focusing more on the users. The teams in the case studies from that paper recognized that using agile methods alone did not identify whether the right product for users was built but going through the build-measure-learn cycle added the needed understanding of the users' and customers' goals and needs.

A review by Adeola Wale-Kolade et al. [12] identified seven claims across the research literature regarding integrating software development and UX design:

- 1 Conduct some upfront design activities before project start.
- 2 Design low-fi prototypes as the basis for developing the system.
- 3 Perform testing between iterations.
- 4 Designers and developers work in parallel.
- 5 Usability designers should be present.
- 6 Usability designers should be fully integrated into the development team.
- 7 End users or their proxies should be involved in the project life cycle

These claims have not escaped criticism. For example, claim 5 should ensure UX designers are present, but with only a simple presence, the role may be filled with other personnel. While claim 6 should ensure that UX design concerns are always present through assimilating UX designers into the development team, UX designers may well identify too closely with the software goals and lose track of UX activities and concerns [27]. There are many other rebuttals to these seven claims, showing that the integration of UX activities with agile development is not simple and that matching several concerns is necessary [12].

2.2. Theoretical framing of agility and integration

Researchers and practitioners ascribing the Agility of a software project to the use of a particular process such as Scrum or Kanban may exacerbate the difficulty of integrating software development and UX design. A team may use a process labeled "Agile" in a way that does not create flexibility, leanness, and adaptability [10]. We see Agility as

referring to software developers' and UX designers' effectiveness in dealing with the inevitable changes that affect the success of a software project. Here, we apply a general understanding of Agility, not tied to a specific process, based on Kieran Conboy's systematic literature review of the concept of Agility across different disciplines [10]. Conboy proposes the following definition of Agility, which emphasizes the core principles of embracing change and providing customer value: "(The) continual readiness ... to rapidly or inherently create change, proactively or reactively embrace change, and learn from change while contributing to perceived customer value (economy, quality, and simplicity) through its collective components and relationships with its environment" [10]. Conboy translated the definition into a formative taxonomy of Agility, stating that to be agile, a process component must:

- 1 contribute to at least one of the following: (i) creation of change, (ii) proaction in advance of change, (iii) reaction to change, (iv) learning from change;
- 2 contribute to at least one of and not detract from the following:
(i)perceived economy, (ii) perceived quality, (iii) perceived simplicity;
- 3 be continually ready, i.e., must require minimal time and cost to prepare the component for use.

We use this definition to investigate how one Danish software company that prominently pursues both UX design and agile software development integrates UX design activities with its software development process. While previous research has used Conboy's theory to distinguish contributions to Agility from project- and firm-level processes [11], we use it here to distinguish between contributions to Agility for UX designers and software developers while still needing to integrate their work.

We define *integration* as a distinct way of coordinating activities in organizational processes. Mintzberg's [28] seminal theory of organizations distinguish five prime coordinating mechanisms: *direct supervision*, *standardization of work processes*, *standardization of skills*, *standardization of outputs*, and *mutual adjustment*. The last-mentioned coordinating mechanism is characteristic of organizations with the structure he calls the *Adhocracy*, often situated in complex and dynamic environments [28]. A well-known situation for software companies striving for Agility. The *Adhocracy* and coordinating through mutual adjustment rely on applying diverse and sophisticated expertise as bases for building new knowledge for innovation. Successful mutual adjustments are the result of informal communication between people conducting interdependent work.

We see mutual adjustment as central to the *integration* of software development and UX design. For further clarity, we distinguish *integration* from *assimilation*, which relies on coordinating by direct supervision and standardization of work processes. Moreover, we see *integration* as different from *separation*, depending on coordination by standardization of skills and outputs, as illustrated in Fig. 1.

Assimilation of agile development and UX design into a process unity, with a single team acting in its own power, has bearings to agile

processes rejecting specialized roles to strive for oneness. Here, the coordination of work involves standardized work processes, shared norms of what is desirable, and direct supervision. In contrast to *assimilation*, *separation* refers to software development and UX design as acting on one another. This twoness has bearings to pre-agile relay races involving the coordination of specialized roles with standardized skills and outputs, e.g., through user studies. *Integration* lies between *assimilation* and *separation*, and it is an across-entities point of view [29] to avoid the reduction into a single entity or the separation into two distinct entities. This *integration* is coordinated by mutual adjustments of the work in the situation at hand. Fig. 1 thus contrasts Mintzberg's [28] coordinating mechanism of mutual adjustment (*integration*) against the four others; two we associate with a single-entity view (*assimilation*) and two with a between-entities view (*separation*). We apply this understanding of *integration* and combine it with Conboy's [10] definition of *Agility* to unfold the relations and social processes [29] of UX design and software development.

3. Method

Our investigation of integrated UX design and software development processes is based on a single case study approach [30,31] to address how these two processes maintain their Agility. A single case study is well suited for studying a contemporary phenomenon in its real-world context [32] to develop insights into our "how" question [33]. This case study design also applies well when the boundary between the phenomenon and the context is unclear, i.e., between the UX design and software development activities and the broader organizational context allowing their Agility. The information-based selection of a single case [32] has the rationale of an unusually [33] high dedication to UX design and agile software development.

3.1. The case

The case setting is a Danish software company, Mjølner Informatics, with 100+ employees who prominently pursue both UX design and agile software development. Mjølner Informatics as a software house develops specialized software products for particular customers on project contracts. Most employees have a master's degree related to UX, interaction design, or software development. The customers are small and large private companies and public organizations. Mjølner Informatics employees may assist customer organizations with specialist knowledge on time-and-materials contracts for up to several months. The Danish edition of *Computerworld* named the company "IT comet of the year, 2015" in Denmark, partly because of its heavy emphasis on UX in software development. UX designers that play crucial and managerial roles in most projects, account for 10% of the employees.

3.2. Data collection

We collected data in two iterations. First, we conducted 10 individual interviews. We interviewed four UX designers, three software

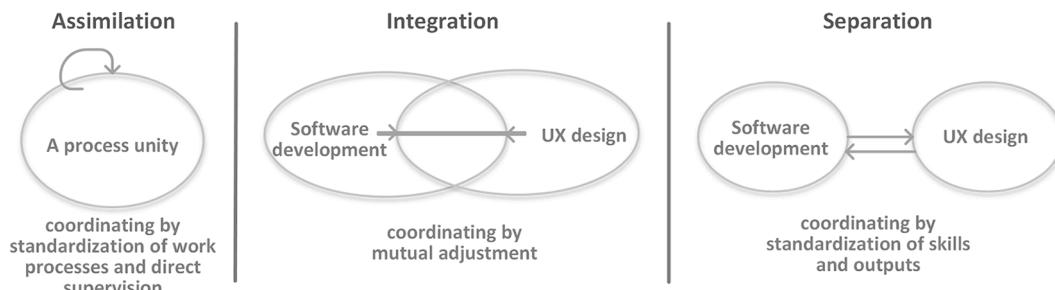


Fig. 1. The three views on the Agility of software development and UX design.

developers, two project managers, and one software architect. Each interview lasted 45–60 min, were audio recorded, and followed Patton's guidelines for pragmatic interviews that *aimed at getting straightforward answers that can yield practical and useful insights* [34]. We had one interviewer and one observer from the research team during each interview. This iteration had a threefold aim of understanding: (1) the agile development process at the company, (2) perceptions of the UX designers' role in different project phases, and (3) the approach of integrating UX design with agile software development.

The second iteration of data collection was conducted two years after the initial interviews. We presented our preliminary findings from the first iteration in a short paper that we shared with the case company during a reflection session with 6 representatives from the case company; three UX designers, two software architects, and one project manager. The project manager also participated in the first iteration two years earlier, while the other participants were new to the study. The software architects also represented the views of software developers as this is the typical career path with an increase in seniority within the company. The session had a two-fold aim of 1) discussing and corroborating our analysis on the integration approach that *was* and 2) discussing the integration approach that *is* two years later to elicit reflections based on our findings presented in Sections 4.1 and 4.2 below. We invited the representatives to reflect on their practices [35], combining our initial interpretation with intervention to better understand the case in its organizational context [36]. Two from the research team facilitated the workshop activity by first providing a recap of our findings from the first iteration of data collection, followed by a discussion. The workshop had a duration of 2 h and was audio recorded.

3.3. Data analysis

We analyzed the data from the 10 individual interviews of the first iteration and the reflection session with 6 participants in the second iteration through a theory-directed qualitative content analysis [37] using Conboy's theory of Agility [10]. Audio recordings from the 10 individual interviews were transcribed in the online tool Dedoose, which we also used to support the content analysis. Two members of the research team conducted the analysis with a particular emphasis on how the effect of integration approaches differs for software developers and UX designers. Audio recordings and notes from the reflection session of the second iteration were analyzed abductively to uncover the participants' views on the integration approach *there was*, *how it is* (two years later), and *how they want* the approach in the future. In Mjølner Informatics, two process elements are essential to integrating UX design with software development in an agile manner: upfront design and work in parallel, which corresponds to the first claim on integrating UX design with an agile development process from Adeola Wale-Kolade et al. [12]: "*Conduct some upfront design activities before project start*" and fourth claim: "*Designers and developers work in parallel*". We present these two in Sections 4.1 and 4.2 and the findings from their workshop reflections on these two in Section 4.3.

4. Findings

4.1. Upfront design

In the upfront design at Mjølner Informatics, the software product is considered from various perspectives before producing any code. Here, UX designers typically participate full-time with responsibilities of defining system requirements, making design sketches, and designing wireframes.

A UX designer in Mjølner Informatics is the main actor collaborating with the customers, the users, and the system architect (an experienced software developer). The system architect and the UX designer collaborate on the validation of wireframes. They consider the system requirements linked to wireframes, which is the primary outcome of the

upfront design. One UX designer argues for this upfront design approach by contrasting it with placing UX design one sprint ahead of software development as follows:

It is incredibly challenging to create wireframes once development has started. As a UX designer, I then need to work one sprint ahead. I have experienced designing wireframes during a sprint, which is very hectic. The results are not good. The analytic part disappears when these activities are done in the sprint. I lose track and overview. I believe that there is a need to work in an upfront manner, as we do. (UX Designer 1)

The same UX designer further explains that the one-sprint-ahead approach can work in some cases, but it is complicated and hard to work that way since all sorts of things happen during the sprint. Smaller details can be designed during software development, but an overview through designing wireframes of the whole system needs to be in place during the upfront design period.

The goal of such upfront design is also relevant to software developers. Another UX designer notes that she makes the design to a level of completeness that allows the developers to take over. She explains:

The overall design could be a graphical design, but also flows where you, for instance, have identified individual elements, which you then specify further in Jira. (UX Designer 2)

When software developers take over, they are aware of the challenges and necessity of UX, but some have limited knowledge of the business aspects and the customer's needs. Upfront design is a challenge because the customer often wants to know the "magical price" of the project, and the software developers need to know more about the business aspects. This need forces UX designers to provide specific details up front through UX design, which they can discuss with the customer. A software developer explains:

At the beginning of the process, there is this "black box" known as UX design, which is typically positioned before the "tech" phase, when developers enter the project. (Developer 1)

Software developers do not perceive their limited upfront involvement and knowledge of what is going on in the project as problematic. Instead, a developer states that he appreciates the defined boundaries of UX designers working up front and developers entering the project later:

I do not know that much about the very first phase, but it is when you define the overall UX, initial research activities, and initial designs. It happens before developers enter the project... It is nice to join the project when the upfront work has been done; there are some defined boundaries. (Developer 2)

However, these defined boundaries do not remove the need for making changes. The same software developer explains that when they get more into development, some tasks that were identified up front no longer fit and have to be changed. He wants to keep the focus on upfront design with detailed wireframes and prototypes. A project manager similarly emphasizes the need for continuous adjustments to a product throughout a project. Adjustments to a product made in one sprint could be made in the next sprint or later in the project. But the initial overview in the upfront design period provides a good idea of what to do, he comments.

As shown from the quotes above, upfront design is not used to standardize and rigidly control the design. Instead, both UX designers and software developers use upfront design to achieve Agility, as summarized in Table 1.

Upfront design helps software developers be *proactive* about changes in the system's architectural design by anticipating development risks (see first point's row in Table 1). This is possible thanks to coordination with UX designers, who readily *react* to specification changes from the very beginning and *learn* from these changes, e.g., through heavy use of design iterations in dialog with customers and users. Upfront design activities support initial reactions to change through a process in which

Table 1

Contributions to Agility with upfront design (based on Conboy's [10] framework in the left column).

Agility	Software development	UX design
1. A process component must contribute to at least one of the following: (i) creation of change (ii) proaction in advance of change (iii) reaction to change (iv) learning from change	Before development, the system architect is proactive about changes and anticipates development risks.	The initial reaction to change and learning from change are strong with intensive customer dialog and no software committed.
2. A process component must contribute to at least one of and not detract from the following: (i) perceived economy (ii) perceived quality (iii) perceived simplicity	Improved simplicity through separation of concerns and less coordination with users.	Perceived quality through intensive communication with users and customers and a modest cost of early changes contribute to the perceived economy.
3. A process component must be continually ready, i.e., require minimal time and cost to prepare the component for use.	A system architect is available and active throughout the project.	The UX designer autonomously controls the project scope.

early visualizations of ideas through, e.g., paper prototyping, facilitate specific discussions on system requirements before committing to any program code or model implementations. UX designers' responsibility to identify requirements by coordinating with customers and users offers software developers *simplicity* with fewer coordination needs. Also, UX designers improve product *quality* through intensive communication with customers and users. At this early stage, changes in requirements and designs incur modest costs, which contribute to the *perceived economy* (defined as the utilization of all resources is maximized, and no unnecessary resources are maintained [10]). Finally, these achievements of Agility are readily available to both software developers and UX designers, mainly through the activities of the software architect and the UX designer (see the third point's row in Table 1).

Thus, upfront design is an example of integration through mutual adjustment between UX design and software development. This also transcends down to the level of specific tools where UX designers also have editing rights in Jira. It contributes to Agility across the two specialized activities without collapsing them into a single activity or process. The contribution to Agility differs for UX designers and software developers at Mjølner Informatics. As an example, the second row in Table 1 shows that the process followed by the software developers (represented by the architect during upfront design) provides simplicity through a separation of concerns as they have less coordination with users. Such coordination is more intensive in the UX design process, which in turn increases perceived quality.

4.2. Work in parallel

The work in parallel approach in Mjølner Informatics involves software developers producing code based on the specifications elicited in the upfront design and UX designers spending about 20% of their time on a project while also working on other projects. The work in parallel period starts when a development team is formed. The team typically includes one UX designer, two to four developers, and one project manager working in a series of iterations, each lasting three to four weeks. The main responsibility of the UX designer during the work in parallel period is to review and sign off implemented designs at the end of each iteration and facilitate product evaluations with customers. The UX designer is physically located close to the software developers for easy access. The UX designer uses the wireframes initially designed in

the upfront design when meeting with the rest of the development team to convey system requirements. A project manager describes this work-in-parallel approach and argues for its necessity for efficient work as follows:

At the beginning of the development stage, the infrastructure is set up: Jira and a Scrum board, wireframes, graphical elements, user stories. It is then up to the team to make the necessary refinements ... If you have two sprints n and n + 1, the UX designer is central in sprint n to prepare sprint n + 1. They need to be ahead so that software developers have some designs to work with; otherwise, it will not be efficient work. (Project Manager 1)

A UX designer continues the project manager's line of reasoning as an aim of her design practice. She explains that while being in one sprint, she needs to begin working on designs for the next sprints. Typically, UX designers would aim to begin one or two sprints ahead. A software developer elaborates on UX designers' responsibilities:

Sometimes the UX designer assumes a reviewing role to go over our implemented designs, and this is the case throughout the project. (Developer 2)

The UX designer acknowledges the role's wide range of responsibilities, which implies limited prerequisite knowledge, making frequent informal communication with developers necessary. She stresses that the development of designs must never become top-down, and there will be some elements that she cannot know. She further explains:

So, we need to engage during development ... I spend about one-sixth of my time during development in projects. During development, we participate in the planning to answer questions from the developers. (UX Designer 2)

A project manager further elaborates UX designers' work in parallel approach. He says that UX designers typically engage full-time in the upfront design of the project, and then their work is phased out. He also comments that daily collaboration with software developers is essential since there will be some minute changes in the design until the final semicolon is set in the code. He further explains:

This works well at Mjølner since the architect, project manager, and UX designer are in the same building or room, so this comes naturally, also as part of the daily Scrum meetings ... The UX designer is part of the team to ensure that requirements are met. Some would think that UX designers create lovely designs and icons, but this is not the central part of that role. (Project Manager 1)

As shown from the quotes above, the work in parallel approach is not aimed at a rigid division of labor or at defending a level of control for particular roles. Instead, both software developers and UX designers use the work in parallel approach to achieve Agility, as summarized in Table 2.

The work-in-parallel approach involves a UX designer meeting with the software developers at the end of each sprint to review implemented designs. This limited time scope of the reviews makes *reactions to change* more manageable for software developers, as they do not have to continuously cope with changes (see first point row in Table 2). On the other hand, this enables the UX designer to focus on *creating change* from the customer perspective. The structured UX input through e.g. specific visualizations of requirements contributes to software developers' *perceived simplicity* and localized efforts at specific points during work-in-parallel contributes to UX designers' *perceived economy* of scale as this enables them to work on multiple projects. Finally, these achievements of Agility are readily available to both software developers and UX designers (see the third point's row in Table 2). Overall, work in parallel is integration through mutual adjustment between UX design and software development, and it contributes to Agility in different ways for UX designers and software developers.

Table 2

Contributions to Agility with work-in-parallel.

Agility	Software development	UX design
1. A process component must contribute to at least one of the following: (i) creation of change (ii) proaction in advance of change (iii) reaction to change (iv) learning from change	UX feedback and design changes limited to sprint reviews make <i>reactions to change</i> more manageable.	Thorough reviews and quality control detached from the development team while in dialog with customers promote the <i>creation of change</i> .
2. A process component must contribute to at least one of and not detract from one of the following: (i) perceived economy (ii) perceived quality (iii) perceived simplicity	<i>Perceived simplicity</i> through structured UXdesign inputs that visualize requirements using e.g. wireframes.	<i>Perceived economy</i> of scale through localized efforts during work-in-parallel enables the ability to work on multiple projects.
3. A process component must be continually ready, i.e., require minimal time and cost to prepare the component for use.	Readily available designs and designers.	Reuse of UX competencies and insights across projects.

In Mjølner Informatics, the two approaches upfront-design and work-in-parallel to integrating UX design with software development contribute to Agility. However, software developers' and UX designers' experiences differ in their dealings with change, simplicity, quality, economy, and readiness.

- Software developers use upfront design to proact in advance of change for the sake of simplicity, whereas UX designers use upfront design to react and learn from change for the sake of quality and economy (Table 1).
- Software developers use work in parallel to respond to change for the sake of simplicity, whereas UX designers use work in parallel to create change for the benefit of an economy of scale (Table 2).

4.3. Reflecting on integration

Our final phase of inquiry revolved around presenting our findings on the integration approach in terms of upfront design and work-in-parallel. We wanted to provide feedback to the case company related to our interpretations using Conboy's theory on Agility and Mintzberg's coordination mechanisms as the analytical lens on their integration. As highlighted in the previous sections, we discovered upfront design and work-in-parallel phases in which software developers and UX designers coordinated through mutual adjustment, representing an approach resembling integration rather than assimilation and separation. We asked the participants to reflect on Mjølner's approach to encompassing agile software development and UX design through an integration approach. We asked them to reflect on how this approach *was* and the extent to which they could recognize it through our analysis. We also asked participants to reflect on how their agile integration approach currently *is* and how they *want* it to look like in the future. Through this reflection, we identified three main points.

4.3.1. Agile integration considered better the way it was

During our inquiry on reflection, it became apparent that Mjølner transitioned into an approach resembling that of separation. One of the architects stated that he was now much less involved in upfront design than three years ago. In his view, the upfront design phase now seemed overly dominated by UX designers.

"Things have happened over the past few years. Customers are no longer willing to pay for upfront activities to the same extent. This means that we now must send only one person, and other times we do not even send one out because the project is defined by the customer already. In that situation it is difficult to say when the upfront design starts because we are not part of the initial exploratory phase. We need to go back to the situation where the UX designer and architect had a high level of collaboration. I want to be able to push back to the UX designer and vice versa to ensure the best solution is developed... I would really like to be part of the process rather than having to interpret others' findings through artifacts in Jira." (Architect 2)

This architect has the impression that there needs to be a more even balance between the UX designers' and architects' responsibilities during up-front design. He found it critical that these roles mutually adjust during this phase for enabling architects to proactively react to change on different technical issues and risks, which are problems outside the typical expertise area and scope of UX designers. In reasoning why the current integration approach is that way, the architect mentioned that the company is experiencing a shift in the market where customers now find it too expensive to assign multiple people to conduct exploratory upfront design activities. In the following, we deal with this point in more detail.

4.3.2. Agile integration depends on the project and customer

Although Mjølner has transitioned into an approach resembling separation, the architects, software developers, and UX designers in some projects integrate by coordinating through mutual adjustments. In such cases, a software architect is more involved during upfront design together with a UX designer. One of the architects stated that this particularly applies when developing software to be used in embedded products.

Additionally, there seems to have been a shift in the role of upfront design due to changing customer needs. This change is closely tied to the point of Mjølner having transitioned into the separation approach. One mentions that as an architect, he is missing the activities that go into fundamentally understanding the domain in which developed systems are supposed to operate. Customers are less willing to pay for such upfront activities. Now, customers are eliciting requirements and designs on their own, on which basis Mjølner is now just supposed to develop a solution. This change is corroborated by one of the UX managers:

"I need to highlight that we are now operating in a different market. Generally speaking, many companies now need an IT department no matter what type of company they are. These IT departments want to solve several of the upfront design issues on their own. This also means that these companies, our customers, now want a higher degree of project ownership on areas that we were responsible for in the past." (UX Manager 2)

As illustrated in the quote above, one of the UX managers now saw a trend for companies to create their own IT departments that will take many exploratory upfront design tasks, which Mjølner was formerly hired to do. This trend influences Mjølner in terms of their UX professionals' ability to initially react to changes and learn from these (first point's row in Table 1), given that the impact through, e.g., customer dialog is reduced with increased fixation of requirements.

4.3.3. Want to transition from separation towards integration

Based on the above observations, Mjølner has transitioned from an integration approach to separation in which software developers and UX designers are now trying to coordinate through standardization of skills and output. One of the UX managers mentioned that the UX designers want the projects to be less person-dependent.:

"I believe, on behalf of the UX designers, that we have mostly worked in an individual manner to create good relations between ourselves and

architects. It is probably time for us to be less dependent on individuals. It should not be about us as UX designers to tell others what we do and what should be done. Rather, it should be about UX designers sitting down with architects and developers to figure out how to collaborate. It's crucial to have a stable set of fixpoints we can orient ourselves towards." (UX Manager 2)

One of the software architects was also explicit on behalf of the software developers in stating that the current separation approach is flawed since the output is not standardized, i.e., the tools used by UX designers during a project are highly dependent on the persons participating. As an alternative, the architect would like to see more widespread use of abstract design tools such as use case diagrams that software developers can relate to. These shared artifacts and design languages enabled coordination through mutual adjustment in the past.

"Some years ago, we had a shared toolbox between UX designers and architects that everyone were able to apply. We had set of shared artifacts and methods that could be used. Currently, we do not have this shared toolbox. The set of artifacts and methods used depends heavily on the persons involved. We want to go back to the past situation where no matter which people we put on projects, they were able to collaborate and knew exactly how to use the tools and methods." (Architect 2)

In summary, Mjølner is currently following an approach of separation, albeit with some challenges in standardizing skills and outputs, but they *want* to transition back into the how it *was*, i.e. integration. They perceived a need for re-obtaining a shared language between UX designers and software developers. Table 3 summarizes our findings from the reflection session in relation to Agility.

In Table 3, the first point's row highlights how the software architects are now less involved during upfront design activities. The architects and UX designers have drifted into a more reactive role with changes in their customer market. This change has shifted the architects and UX designers into a position of reacting to customers' specifications compared to the previously more collaborative and creative nature of their roles two years prior. The second point's row of Table 3 shows how simplicity and economy are now perceived differently than two years earlier, which in the way of working is perceived by the architects and UX designers as potentially having negative consequences on quality. The third point's row in Table 3 outlines how the process components in the current situation are less readily available, partly due to not being involved during upfront design activities, but in particular also through the challenge of individual dependencies and not having a shared design language and toolbox. Overall, Table 3 outlines how their Agility has become more different and separated, making their practices mutually compete for Agility. This situation is distinctively different from the findings from two years earlier (c.f. Tables 1 and 2), where they integrate their activities by mutual adjustment. Moreover, it shows that integrating by mutual adjustments is itself susceptible to change and that reflecting on integration helps revisit their state of agile and point out opportunities for changing their situation.

5. Discussion

Our study aimed to address the research question: *How can integrated UX design and software development processes maintain their Agility?* Hence, we presented a case study of the integrated UX design and software development processes at the Danish software company Mjølner Informatics. This section discusses how our study contributes to the extant research, its implications for practice, limitations, and future research.

5.1. Contributions to research

Two concepts were central to our case study of agile software development and UX design at Mjølner Informatics. The first was *Agility*,

Table 3

Contributions to Agility as is in relation to upfront design and work-in-parallel.

Agility	Software development	UX design
1. A process component must contribute to at least one of the following: (i) creation of change (ii) proaction in advance of change (iii) reaction to change (iv) learning from change	Upfront Design Involved in some projects, but not all. Challenging to be proactive, no shared design language between software developers and UX designers. Work in Parallel Drift from reacting to changes through collaboration with UX design towards reacting to handovers from customers.	Upfront Design Reactive to changes and learning from change, but not in all projects. Tool use depends on the individual. Work in Parallel Drift from a role as creator of change to more simple reactions to changes specified by customers.
2. A process component must contribute to at least one of and not detract from one of the following: (i) perceived economy (ii) perceived quality (iii) perceived simplicity	Upfront Design Perceived economy due to not participating in upfront design activities comes at the cost of perceived quality. Work in Parallel Perceived simplicity through pre-made specifications from customers and UX designers comes at the cost of perceived quality and economy because some specifications are challenging to implement, but also because UX output does not provide a shared design language.	Upfront Design Perceived simplicity through not having to coordinate with an architect. In some cases, the customer conduct upfront design, which contribute to perceived simplicity and economy, but at the cost of perceived quality. Work in Parallel Perceived simplicity through pre-made specifications from customers (in some projects) comes at the cost of perceived economy and quality by implementation challenges.
3. A process component must be continually ready, i.e., require minimal time and cost to prepare the component for use.	Upfront Design Software developers and system architects are not always present during upfront design activities. Work in Parallel Readily available developers and designers. Designs are not based on a shared design language nor based on mutual adjustments and are therefore not readily available.	Upfront Design The UX designer is readily available, not in all projects, however, due to customer autonomy in creating specifications. Work in Parallel UX competencies and insight may not necessarily be reused across projects due to not having a shared design language.

for which we used Conboy's [10] taxonomy that allowed us to be open to the meaning of Agility in our specific case rather than judging it according to some of the many methodologies available in the literature (e.g., [13,14,20,21]). The second was *integration*, which we defined as involving mutual adjustments based on Mintzberg's classical theory of organizations [28] and thereby differentiated *integration* from other ways of coordinating through *standardization of work processes, skills, and outputs*, or *direct supervision*. From analyzing our case, we show how Agility may differ for UX designers and software developers when integrating their efforts in a software project through upfront design (cf. Section 4.1) and work in parallel (cf. Section 4.2). This finding contributes to previous research on the challenges of integrating UX into agile processes [14] by showing how the ideals of successful integration can be specific to the role and situation.

Next, we showed in our workshop (cf. Section 4.3) that these concepts also were helpful to the practitioners for reflecting on what *was*, presently *is*, and what they *want* for their integration. The conceptualizations provided a starting point for deliberation on their practices in the workshop. While this starting point for deliberation is more abstract by

focusing on the overall integration process than previous research on, e.g., concise user stories [8] or technical debt items [38], it still allows positioning and discussing *Agility* in their specific situation. We use the three views on the *Agility* of software development and UX design (cf. Fig. 1) to explain our case study at Mjølner Informatics and as a basis for proposing the three distinct claims about *Agility* presented in Fig. 2.

On the left in Fig. 2, we have *assimilation* that involves coordinating by standardization of work processes and direct supervision, which potentially makes *Agility* for UX and software to be mutually indistinguishable. Remnants of this view were present in our case, as UX has been and still is a concern for the software developers and goes back to agile methods such as Scrum [39] and extreme programming [40] that were skeptical of technical roles beyond that of a team member.

The *integration* view that involves coordinating by mutual adjustment, which potentially makes *Agility* for UX and software to be different and mutually complementary, was a central focus in our case study. Our findings from Mjølner Informatics' (cf. Sections 4.1 and 4.2) substantiated this claim in practice. However, as shown in a workshop two years after our first interviews, this *integration* is fragile and may drift towards *separation*, as shown on the right in Fig. 2.

The *separation* view, which involves coordinating by standardization of skills and outputs, potentially makes *Agility* for UX and software to be different and mutually competing. Our case study found that changes in Mjølner Informatics' market (what Mintzberg calls a situational factor [28]) further pushed the software developers and UX designers towards *separation*. This market orientation shifted some of the UX design activities into the customer organization, which made it difficult for Mjølner Informatics' internal UX designers to adjust to the concerns of the software developers. The software developers disliked this *separation* from UX and requested they return to integration at our workshop. However, a single case study like ours cannot claim that any of the three is superior to the two others, only that they are feasible and that UX design and software development processes may transition between them over time.

Overall, Fig. 2 distinguishes three relationship types based on Mintzberg's theory of organizations [28] as having inherent views on *Agility*. To unfold these views, Tables 1 and 2 (cf. Sections 4.1 and 4.2) are exemplars for analyzing the specific contributions to *Agility* in a software development and UX design relationship on a more detailed level. Table 3 (cf. Section 4.3) further shows how Conboy's [10] taxonomy can be useful for reflecting on maintaining *Agility* according to the three claims in Fig. 2.

5.2. Implications for practice

Our case study has some practical implications, and we propose a three-step inquiry to help practitioners understand integration in a specific situation or project to realize these implications. The first step is to identify approaches to integrating UX design with software development. Here, a project manager or someone dedicated to facilitating

Agility, such as a Scrum master, may prefer assimilation or separation over integration (see Fig. 2). A preference for assimilation may be rooted in the team-centric principles in the agile manifesto [2] or Scrum guide [41] for the all-inclusive label of developers to be collectively accountable for their work. In contrast, a preference for separation may be rooted in the distinctions of roles and tasks from frameworks such as the Unified Process [42]. The project manager or Scrum master can identify approaches to integration using the previously discussed seven claims regarding agility and UX design [12]. However, these seven claims are not an exhaustive list; other approaches may be more specific to the situation. The second step is to assess the contributions to *Agility* for both software development and UX design. This assessment considers a process component's contribution to managing change, perceived economy, quality, and simplicity, and its continual readiness for use [10]. Tables 1–3 present examples of such an assessment. The third step involves reflecting on their mutual adjustments inherent to these assessments of the situation in order to improve *Agility*. Our study shows (cf. the workshop presented in Section 4.3) how this reflection is useful for uncovering an unwanted drift toward separation; thus, we believe that revisiting these three steps also can help proactively avoid it.

5.3. Limitations

Our single case study of UX design and software development processes provides evidence of what is feasible for their integration and *Agility*. We can not make any claims on what is preferable or effective on a generalized level. Unfolding the concepts of *Agility* and *Integration* with detailed empirical insights from a single case contributes to richness rather than representation. Our case was not chosen to be representative of agile software development and UX design integration in most cases but as an unusual case [32] with a Danish organization highly dedicated to both UX and *Agility*. This information-oriented case selection for integration of *Agility* and UX Design implies analytical rather than a statistical generalization. Our case study's analytic generalization advances theoretical concepts [33], specifically the theory of *Agility*, to consider the practices of upfront design and work in parallel as potential contributions to the *Agility* of UX design and software development. This finding, although limited in terms of statistical generalizability, is interesting because, according to earlier research, these two practices detract from *Agility* in software development.

5.4. Future research

Our study points to ample opportunities for future research that compares the effect on *Agility* from different approaches to integration. Other researchers may conduct comparative case studies or surveys of multiple organizations to determine such effects. Our findings from an unusual case set in an organization highly dedicated to both UX and *Agility* may be empirically tested by comparison with a representative

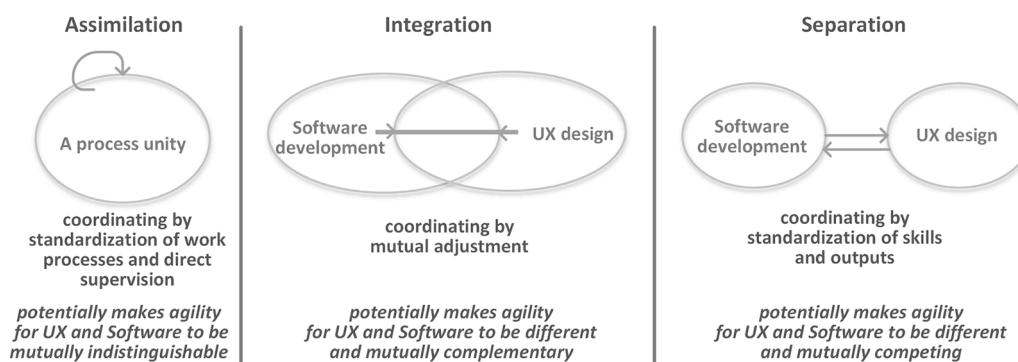


Fig. 2. Three claims about *Agility* for software development and UX design.

sample of organizations' integration practices. Moreover, we found that situational factors [28], such as the organization's market environment, may influence the coordination and integration of UX design and software development processes. For organizations, situational factors may include their age and size, technical systems, environment, and power structures [28], which could be important for explaining how they integrate UX design and software development processes. Finally, future research may also test the usefulness and transferability of our findings through action research, similar to previous efforts on the challenges of integrating UX work with agile software development [8]. A starting point could be to test and theoretically elaborate the three steps outlined in the previous section on implications for practice.

6. Conclusion

To answer how integrated UX design and software development processes can maintain their Agility, we present a case study at Mjølner Informatics', a company highly dedicated to UX design and agile software development. We analyzed the company's integration approaches of upfront design and work in parallel with Conboy's taxonomy of Agility [10]. This analysis showed how Agility differs for the two roles with these two integration approaches. They experienced notably different work processes that still dealt effectively with change and contributed to quality, economy, or simplicity. We explain that their integration through mutual adjustment makes the Agility for UX designers and software developers different yet complementary. This integration contrasts with assimilation, which potentially makes their Agility mutually indistinguishable, and with separation, which makes their Agility different and mutually competing.

Our follow-up workshop two years after our first interviews also showed that the processes of upfront design and work in parallel were susceptible to integration struggles over time. At that point, we found a drift towards separation, making their Agility increasingly different and mutually competing. This finding suggests practitioners should reflect more frequently on how their integration approaches afford Agility and to whom.

CRediT authorship contribution statement

John Stouby Persson: Conceptualization, Methodology, Visualization, Formal analysis, Investigation, Writing – original draft. **Anders Bruun:** Formal analysis, Investigation, Writing – original draft, Project administration. **Marta Kristín Lárusdóttir:** Investigation, Writing – original draft, Project administration. **Peter Axel Nielsen:** Writing – review & editing, Visualization.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

- [1] A. Bruun, M.K. Larusdottir, L. Nielsen, et al., The role of UX professionals in agile development: a case study from industry, in: G. Berget (Ed.), Proceedings of the 10th Nordic Conference on Human-Computer Interaction, ACM, 2018, pp. 352–363.
- [2] Beck, K., Beedle, M., Van Bennekum, A. et al.: Manifesto for Agile Software Development. (2001).
- [3] C. Larman, V.R. Basili, Iterative and incremental developments. A brief history, Computer 36 (2003) 47–56 (Long Beach Calif).
- [4] Digital.ai: The 15th Annual State of Agile Report (<https://Digital.Ai/Resources/State-of-Agile>). (2021).
- [5] S. Balaban, J. Durašković, Agile project management as an answer to changing environment, Eur. Proj. Manag. J. 11 (2021) 12–19.
- [6] D.Ø. Madsen, The evolutionary trajectory of the agile concept viewed from a management fashion perspective, Soc. Sci. 9 (2020) 69.
- [7] M. Larusdottir, J. Gulliksen, Å. Cajander, A license to kill—improving UCSD in agile development, J. Syst. Softw. 123 (2017) 214–222.
- [8] A. Ananjeva, J.S. Persson, A. Bruun, Integrating UX work with agile development through user stories: an action research study in a small software company, J. Syst. Softw. 170 (2020), 110785.
- [9] K. Schmitz, R. Mahapatra, S. Nerur, User engagement in the era of hybrid agile methodology, IEEE Softw. 36 (2018) 32–40.
- [10] K. Conboy, Agility from first principles: reconstructing the concept of agility in information systems development, Inf. Syst. Res. 20 (2009) 329–354.
- [11] J.S. Persson, J. Nørberg, P.A. Nielsen, Improving ISD agility in fast-moving software organizations, Anonymous, in: Proceedings of the 24th European Conference on Information Systems, AIS, İstanbul, Turkey, 2016, pp. 1–16.
- [12] A. Wale-Kolade, P.A. Nielsen, T. Päiväranta, Usability work in agile systems development practice: a systematic review. Anonymous Building Sustainable Information Systems, Springer, 2013, pp. 569–582.
- [13] E.L. Law, M.K. Lárusdóttir, Whose experience do we care about? Analysis of the fitness of scrum and kanban to user experience, Int. J. Hum. Comput. Interact. 31 (2015) 584–602.
- [14] D. Argumanis, A. Moquillaza, F. Paz, Challenges in integrating SCRUM and the user-centered design framework: a systematic review, in: V. Agredo-Delgado, K. O. Villalba-Condori, P.H. Ruiz (Eds.), Iberoamerican Workshop on Human-Computer Interaction, Springer, 2020, pp. 52–62.
- [15] K. Curcio, R. Santana, S. Reinehr, et al., Usability in agile software development: a tertiary study, Comput. Stand. Interfaces 64 (2019) 61–77.
- [16] T.S. Da Silva, M.S. Silveira, F. Maurer, et al., The evolution of agile UXD, Inf. Softw. Technol. 102 (2018) 1–5.
- [17] D. Teka, Y. Dittrich, M. Kifle, Adapting lightweight user-centered design with the scrum-based development process, Anonymous, in: Proceedings of the ACM/IEEE Symposium on Software Engineering in Africa, 2018, pp. 35–42.
- [18] D. Sy, Adapting usability investigations for agile user-centered design, J. Usability Stud. 2 (2007) 112–132.
- [19] T.S. Da Silva, A. Martin, F. Maurer, et al., User-centered design and agile methods: a systematic review, Anonymous, in: Proceedings of the Agile Conference, IEEE, 2011, pp. 77–86.
- [20] M. Brehl, H. Meth, A. Maedche, et al., Exploring principles of user-centered agile software development: a literature review, Inf. Softw. Technol. 61 (2015) 163–181.
- [21] D. Salah, R.F. Paige, P. Cairns, A systematic literature review for agile development processes and user centred design integration, Anonymous, in: Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering, ACM, 2014, pp. 1–10.
- [22] A.P.O. Bertholdo, F. Kon, M.A. Gerosa, Agile usability patterns for user-centered design final stages, in: M. Kurosu (Ed.), Proceedings of the International Conference on Human-Computer Interaction, Springer, 2016, pp. 433–444.
- [23] K. Kuusinen, T. Mikkonen, S. Pakarinen, Agile user experience development in a large software organization: good expertise but limited impact, Anonymous, in: Proceedings of the International Conference on Human-Centred Software Engineering, Springer, 2012, pp. 94–111.
- [24] J. Pilz, J. Deutschländer, J. Thomaschewski, et al., Integrating agile human-centered design with lean UX and scrum, Anonymous, in: Proceedings of the 17th International Conference on Web Information Systems and Technologies, 2021, pp. 467–473.
- [25] I. Signoretti, S. Marczał, L. Salerno, et al., Boosting agile by using user-centered design and lean startup: a case study of the adoption of the combined approach in software development, Anonymous, in: Proceedings of the ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), IEEE, 2019, pp. 1–6.
- [26] M. Zorzetti, I. Signoretti, L. Salerno, et al., Improving agile software development using user-centered design and lean startup, Inf. Softw. Technol. 141 (2022), 106718.
- [27] M. Detweiler, Managing UCD within agile projects, Interactions 14 (2007) 40–42.
- [28] H. Mintzberg, Mintzberg on Management: Inside our Strange World of Organizations, The Free Press, New York, NY, 1989.
- [29] C. Morgner, Reinventing social relations and processes:john dewey and transactions, in: C. Morgner (Ed.), John Dewey and the Notion of Trans-action, Springer, 2020, pp. 1–30.
- [30] P. Runeson, M. Höst, Guidelines for conducting and reporting case study research in software engineering, Empir. Softw. Eng. 14 (2009) 131–164.
- [31] J. Lazar, J.H. Feng, H. Hochheiser, Research Methods in Human-Computer Interaction, 2nd ed., Morgan Kaufmann, Cambridge, MA, 2017.
- [32] B. Flyvbjerg, Five misunderstandings about case-study research, Qual. Inq. 12 (2006) 219–245.
- [33] R.K. Yin, Case Study Research and applications: Design and Methods, 6th ed., 5, Sage Publications Inc, 2018.
- [34] M.Q. Patton, Qualitative Research & Evaluation methods: Integrating Theory and Practice, Sage Publications, 2015.
- [35] D.A. Schön, The Reflective Practitioner: How Professionals Think in Action, Basic Books, New York, NY, 1983.
- [36] K. Braa, R. Vidgen, Interpretation, intervention, and reduction in the organizational laboratory: a framework for in-context information system research, Account. Manag. Inf. Technol. 9 (1999) 25–47.
- [37] H. Hsieh, S.E. Shannon, Three approaches to qualitative content analysis, Qual. Health Res. 15 (2005) 1277–1288.
- [38] N.B. Borup, A.L.J. Christiansen, S.H. Tovgaard, et al., Deliberative technical debt management: an action research study, in: X. Wang, A. Martini, A. Nguyen-Duc, et al. (Eds.), Proceedings of the 12th International Conference on Software Business, Springer, 2021, pp. 50–65.

- [39] K. Schwaber, Scrum development process, Anonymous. *Business Object Design and Implementation*, Springer, 1997, pp. 117–134.
- [40] Beck, K.: Extreme programming explained: embrace change. addison-wesley professional (2000).
- [41] Schwaber, K., & Sutherland, J.: The scrum guide: the definitive guide to scrum: the rules of the game (2020).
- [42] C. Larman, *Agile and Iterative Development: A Manager's Guide*, Pearson Education Inc., Boston, MA, 2004.

Embracing Change with Extreme Programming



Extreme Programming turns the conventional software process sideways. Rather than planning, analyzing, and designing for the far-flung future, XP programmers do all of these activities—a little at a time—throughout development.

Kent Beck

First Class
Software

In the beginning was the waterfall (Figure 1a). We would get the users to tell us once and for all exactly what they wanted. We would design the system that would deliver those features. We would code it. We would test to make sure the features were delivered. All would be well.

All was not well. The users didn't tell us once and for all exactly what they wanted. They didn't know. They contradicted themselves. They changed their minds. And the users weren't the only problem. We programmers could think we were making great progress only to discover three-fourths of the way through that we were one-third of the way through.

If long development cycles were bad because they couldn't adapt to changes, perhaps what we needed was to make shorter development cycles. As Figure 1b

shows, the waterfall begat iterations.

The waterfall model didn't just appear. It was a rational reaction to the shocking measurement that the cost of changing a piece of software rose dramatically over time. If that's true, then you want to make the biggest, most far-reaching decisions as early in the life cycle as possible to avoid paying big bucks for them.

The academic software engineering community took the high cost of changing software as a challenge, creating technologies like relational databases, modular programming, and information hiding. What if all that hard work paid off? What if we got good at reducing the costs of ongoing changes? What if we didn't have to settle for taking a cleaver to the waterfall? What if we could throw it in a blender?

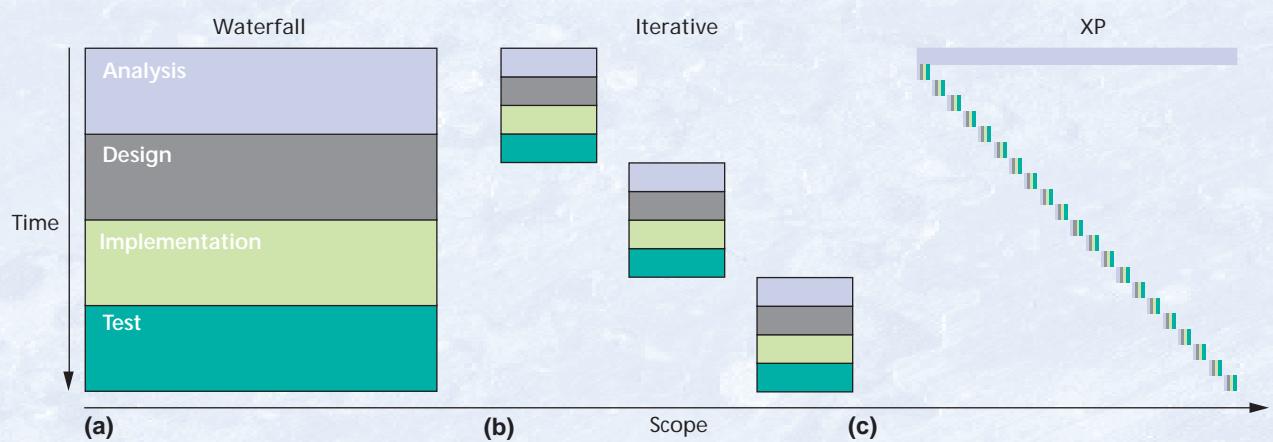


Figure 1. The evolution of the Waterfall Model (a) and its long development cycles (analysis, design, implementation, test) to the shorter, iterative development cycles within, for example, the Spiral Model (b) to Extreme Programming's (c) blending of all these activities, a little at a time, throughout the entire software development process.

XP Practices

Here is a quick summary of each of the major practices in XP.

Planning game. Customers decide the scope and timing of releases based on estimates provided by programmers. Programmers implement only the functionality demanded by the stories in this iteration.

Small releases. The system is put into production in a few months, before solving the whole problem. New releases are made often—anywhere from daily to monthly.

Metaphor. The shape of the system is defined by a metaphor or set of metaphors shared between the customer and programmers.

Simple design. At every moment, the design runs all the tests, communicates everything the programmers want to communicate, contains no

duplicate code, and has the fewest possible classes and methods. This rule can be summarized as, “Say everything once and only once.”

Tests. Programmers write unit tests minute by minute. These tests are collected and they must all run correctly. Customers write functional tests for the stories in an iteration. These tests should also all run, although practically speaking, sometimes a business decision must be made comparing the cost of shipping a known defect and the cost of delay.

Refactoring. The design of the system is evolved through transformations of the existing design that keep all the tests running.

Pair programming. All production code is written by two people at one screen/keyboard/mouse.

Continuous integration. New code is integrated with the current system after no more than a few hours.

When integrating, the system is built from scratch and all tests must pass or the changes are discarded.

Collective ownership. Every programmer improves any code anywhere in the system at any time if they see the opportunity.

On-site customer. A customer sits with the team full-time.

40-hour weeks. No one can work a second consecutive week of overtime. Even isolated overtime used too frequently is a sign of deeper problems that must be addressed.

Open workspace. The team works in a large room with small cubicles around the periphery. Pair programmers work on computers set up in the center.

Just rules. By being part of an Extreme team, you sign up to follow the rules. But they’re just the rules. The team can change the rules at any time as long as they agree on how they will assess the effects of the change.

We might get a picture like the one shown in Figure 1c. It’s called Extreme Programming.

ANATOMY OF XP

XP turns the conventional software process sideways. Rather than planning, analyzing, and designing for the far-flung future, XP exploits the reduction in the cost of changing software to do all of these activities a little at a time, throughout software development. (The “XP Practices” sidebar will give you a quick grasp of the practices and philosophy underlying XP. These practices are designed to work together, and trying to examine any one soon leads you to the rest. The “Roots of XP” sidebar on page 73 traces the historical antecedents of these practices.)

XP development cycle

Figure 2 shows XP at timescales ranging from years to days. The customer picks the next release by choosing the most valuable features (called *stories* in XP) from among all the possible stories, as informed by the costs of the stories and the measured speed of the team in implementing stories.

The customer picks the next iteration’s stories by choosing the most valuable stories remaining in the release, again informed by the costs of the stories and the team’s speed. The programmers turn the stories into smaller-grained tasks, which they individually

accept responsibility for.

Then the programmer turns a task into a set of test cases that will demonstrate that the task is finished. Working with a partner, the programmer makes the test cases run, evolving the design in the meantime to maintain the simplest possible design for the system as a whole.

Stories

XP considers the period before a system first goes into production to be a dangerous anomaly in the life of the project and to be gotten over as quickly as possible. However, every project has to start somewhere.

The first decisions to make about the project are what it could do and what it should do first. These decisions are typically the province of analysis, hence the thin blue analysis rectangle at the top of Figure 1c. You can’t program until you know what you’re programming.

You put the overall analysis together in terms of stories, which you can think of as the amount of a use case that will fit on an index card. Each story must be business-oriented, testable, and estimable.

A month is a good long time to come up with the stories for a 10 person-year project. It’s true that it isn’t enough to explore all of the possible issues thoroughly. But forever isn’t long enough to explore all of the issues thoroughly if you never implement.

Release

Notice in Figure 2 that we don't implement all of the stories at first. Instead, the customer chooses the smallest set of the most valuable stories that make sense together. First we implement those and put them into production. After that we'll implement all the rest.

Picking the scope for a release is a little like shopping for groceries. You go to the store with \$100 in your pocket. You think about your priorities. You look at the prices on the items. You decide what to buy.

In the planning game (the XP planning process), the items are the stories. The prices are the estimates on the stories. The budget is calculated by measuring the team's output in terms of estimated stories delivered per unit time.

The customer can either load up a cart (pick a set of stories) and have the programmers calculate the finish date or pick a date and have the programmers calculate the budget, then choose stories until they add up.

Iteration

The goal of each iteration is to put into production some new stories that are tested and ready to go. The process starts with a plan that sets out the stories to be implemented and breaks out how the team will accomplish it. While the team is implementing, the customer is specifying functional tests. At the end of the iteration, the tests should run and the team should be ready for the next iteration.

Iteration planning starts by again asking the customer to pick the most valuable stories, this time out

of the stories remaining to be implemented in this release. The team breaks the stories down into tasks, units of implementation that one person could implement in a few days. If there are technical tasks, like upgrading to a new version of a database, they get put on the list too.

Next, programmers sign up for the tasks they want to be responsible for implementing. After all the tasks are spoken for, the programmer responsible for a task estimates it, this time in ideal programming days. Everyone's task estimates are added up, and if some programmers are over and some are under, the under-committed programmers take more tasks.

Over the course of the iteration, the programmers implement their tasks. As they complete each task, they integrate its code and tests with the current system. All tests must run or the code cannot be integrated.

As the customer delivers the functional tests during the iteration, they are added to the suite. At the end of the iteration, all unit tests and all functional tests run.

Task

To implement a task, the responsible programmer first finds a partner because all production code is written with two people at one machine. If there is any question about the scope or implementation approach, the partners will have a short (15-minute) meeting with the customer and/or with the programmers most knowledgeable about the code most likely to be touched during implementation.

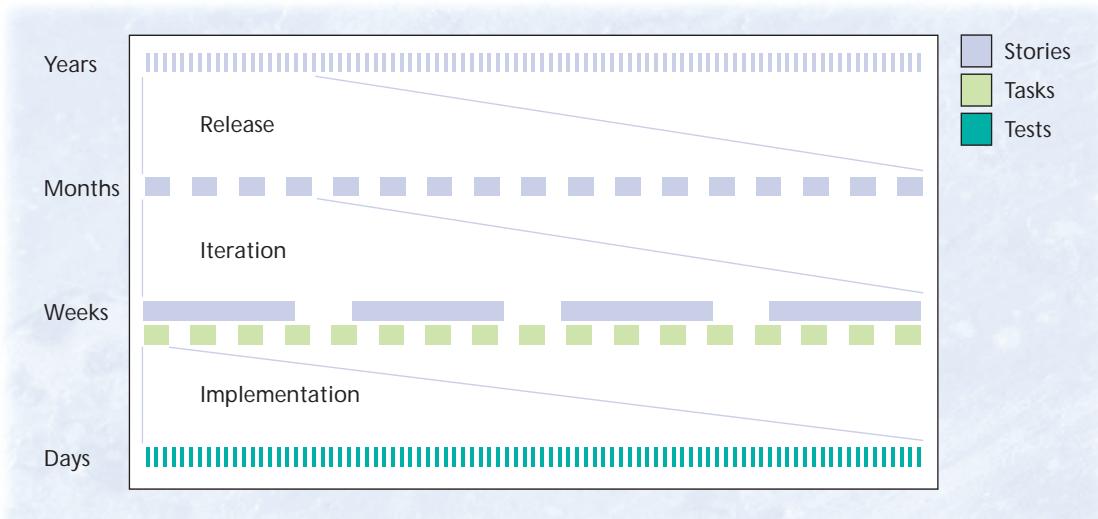


Figure 2. XP according to various timescales. At the scale of months and years, you have the stories in this release and then the stories in future releases. At the scale of weeks and months, you have stories in this iteration and then the stories remaining in this release. At the scale of days and weeks, you have the task you are working on now and then the rest of the tasks in the iteration. And at the scale of minutes and days, you have the test case you are working on now and then the rest of the test cases that you can imagine.

Roots of XP

The individual practices in XP are not by any means new. Many people have come to similar conclusions about the best way to deliver software in environments where requirements change violently.¹⁻³

The strict split between business and technical decision making in XP comes from the work of the architect Christopher Alexander, in particular his work *The Timeless Way of Building*,⁴ where he says that the people who occupy a structure should (in conjunction with a building professional) be the ones to make the high-impact decisions about it.

XP's rapid evolution of a plan in response to business or technical changes echoes the Scrum methodology⁵ and Ward Cunningham's Episodes pattern language.⁶

The emphasis on specifying and scheduling projects from the perspective of features comes from Ivar Jacobson's work on use cases.⁷

Tom Gilb is the guru of evolutionary delivery. His recent writings on EVO⁸ focus on getting the software into production in a matter of weeks, then growing it from there.

Barry Boehm's Spiral Model was the initial response to the waterfall.⁹ Dave

Thomas and his colleagues at Object Technology International have long been champions of exploiting powerful technology with their JIT method.¹⁰

XP's use of metaphors comes from George Lakoff and Mark Johnson's books, the latest of which is *Philosophy in the Flesh*.¹¹ It also comes from Richard Coyne, who links metaphor with software development from the perspective of postmodern philosophy.¹²

Finally, XP's attitude toward the effects of office space on programmers comes from Jim Coplien,¹³ Tom DeMarco, and Tim Lister,¹⁴ who talk about the importance of the physical environment on programmers.

References

1. J. Wood and D. Silver, *Joint Application Development*, John Wiley & Sons, New York, 1995.
2. J. Martin, *Rapid Application Development*, Prentice Hall, Upper Saddle River, N.J., 1992.
3. J. Stapleton, *Dynamic Systems Development Method*, Addison Wesley Longman, Reading, Mass., 1997.
4. C. Alexander, *The Timeless Way of Building*, Oxford University Press, New York, 1979.
5. H. Takeuchi and I. Nonaka, "The New Product Development Game," *Harvard Business Rev.*, Jan./Feb. 1986, pp. 137-146.
6. W. Cunningham, "Episodes: A Pattern Language of Competitive Development," *Pattern Languages of Program Design 2*, J. Vlissides, ed., Addison-Wesley, New York, 1996.
7. I. Jacobsen, *Object-Oriented Software Engineering*, Addison-Wesley, New York, 1994.
8. T. Gilb, *Principles of Software Engineering Management*, Addison-Wesley, Wokingham, UK, 1988.
9. B. Boehm, "A Spiral Model of Software Development and Enhancement," *Computer*, May 1988, pp. 61-72.
10. D. Thomas, "Web Time Software Development," *Software Development*, Oct. 1998, p. 80.
11. G. Lakoff and M. Johnson, *Philosophy in the Flesh*, Basic Books, New York, 1998.
12. R. Coyne, *Designing Information Technology in the Postmodern Age*, MIT Press, Cambridge, Mass., 1995.
13. J.O. Coplien, "A Generative Development Process Pattern Language," *The Patterns Handbook*, L. Rising, ed., Cambridge University Press, New York, 1998, pp. 243-300.
14. T. DeMarco and T. Lister, *Peopleware*, Dorset House, New York, 1999.

From this meeting, the partners condense the list of test cases that need to run before the task is done. They pick a test case from the list that they are confident they can implement and that will teach them something about the task. They code up the test case. If the test case already runs, they go on. Normally, though, there is work to be done.

When we have a test case and it doesn't run, either

- we can see a clean way to make it run, in which case we make it run; or
- we can see an ugly way to make it run, but we can imagine a new design in which it could be made to run cleanly, in which case we refactor the system to make it run cleanly; or
- we can see an ugly way to make it run, but we can't imagine any refactoring, in which case we make it run the ugly way.

After the test case runs, if we see how to refactor

the system to make it even cleaner, we do so.

Perhaps during the implementation of this test case we imagine another test case that should also run. We note the new test case on our list and continue. Perhaps we spot a bigger refactoring that doesn't fit into the scope of our current test. We also note that and continue. The goal is to remain focused so we can do a good job and at the same time not lose the benefits of the insights that come during intense interaction with code.

Test

If there is a technique at the heart of XP, it is unit testing. As you saw above, unit testing is part of every programmer's daily business. In XP, however, two twists on conventional testing strategies make tests far more effective: Programmers write their own tests and they write these tests before they code. If programming is about learning, and learning is about getting lots of feedback as quickly as possible, then you

can learn much from tests written by someone else days or weeks after the code. XP primarily addresses the accepted wisdom that programmers can't possibly test their own code by having you write code in pairs.

Some methodologies, like Cleanroom,¹ prohibit programmers testing or in some cases even compiling their own programs. The usual process has a programmer write some code, compile it, make sure it works, then pass it on to a testing organization. The bench testing takes the form of single-stepping through the code and watching variables, or interpreting the results of print statements, or poking a few buttons to make sure the list item turns green.

The XP testing strategy doesn't ask any more work than the usual bench testing strategies. It just changes the form of the tests. Instead of activities that evaporate into the ether as soon as they are finished, you record the tests in a permanent form. These tests will run automatically today, and this afternoon after we all integrate, and tomorrow, and next week, and next

year. The confidence they embody accumulates, so an XP team gains confidence in the behavior of its system over time.

As I mentioned earlier, tests also come from the customers. At the beginning of an iteration, the customers think about what would convince them that the stories for an iteration are completed. These thoughts are converted into systemwide tests, either directly by the customer using a textual or graphical scripting language or by the programmers using their own testing tools. These tests, too, accumulate confidence, but in this case they accumulate the customer's confidence in the correct operation of the system.

WHEN SOMETHING GOES WRONG

Talking about how a method works when it works perfectly is about like describing precisely how you will descend a monstrous patch of white water. What is interesting is precisely what you will do when the unexpected or undesired happens. Here are some common failures and possible Extreme reactions.

Acxiom: Working toward a Common Goal

Jim Hannula, Acxiom

On top of a data warehouse, Acxiom built a campaign management application using Forté's distributed OO development tool. The small development team—consisting of 10 developers—built the application by relying on sound OO principles and a strong team development approach.

During the final two years of the application's three years of development, the team—comprised of managers, business analysts, developers, testers, and technical writers—used Extreme Programming techniques, which proved to be instrumental in our success.

We know we have a good design if it's simple. Some of our past designs tried even to account for future iterations of our application. We discovered that we were not very good at that. If we use patterns and communicate well, we can develop a sound application that is flexible and can still be modified in the future.

Refactoring is a major part of our development effort. It was evident to us that if we were afraid to change some code because we did not know what it

did, we were not very good developers. We were letting the code control us. If we don't know what the code does now, we break it and find out. It is better to implement a solid piece of code than it is to let a piece of code control the application.

Unit testing was a hard piece to implement because Forté did not have a ready-built testing framework. We developed our own testing framework and have been successful implementing it. Recently we started using Java as a development language and now use JUnit as a testing tool.

The key to XP is setting developer and team expectations. We have found all developers on the team must buy into Extreme or it doesn't work. We tell prospective developers if they do not want to follow our development style, this is not a good team for them. One person not buying in to the approach will bring down the whole team. XP focuses on the team working together to come up with new ideas to develop the system.

When we first started with XP, some of the developers did not want to follow it. They felt that it would hurt their development style and that they would not be as productive. What happened was that their pieces of the application were producing the most problem reports. Since



Team: managers, business analysts, developers, testers, and technical writers

Application: campaign management dbase

Time: three years

they were not developing in pairs, two people had not designed the subsystem and their skills were falling behind the other developers who were learning from each other. Two well-trained developers working together and with the rest of the team will always outperform one "intelligent" developer working alone.

A misconception about XP is that it stifles your creativity and individual growth. It's actually quite the contrary. XP stimulates growth and creativity and encourages team members to take chances. The key is to decide the direction of the corporation and stand behind the hard decisions.

XP is not extreme to our team. It's a method that uses a common-sense development approach. Everyone works together toward a common goal.

DaimlerChrysler: The Best Team in the World

Chet Hendrickson, DaimlerChrysler

The C3 project began in January 1995 under a fixed-priced contract that called for a joint team of Chrysler and contract partner employees. Most of the development work had been completed by early 1996. Our contract partners had used a very GUI-centered development methodology, which had ignored automated testing. As a result, we had a payroll system that had a lot of very cool GUIs, calculated most employees' pay incorrectly, and would need about 100 days to generate the monthly payroll. Most of us knew the program we had written would never go into production.

We sought Kent Beck to help with performance tuning. He found what he had often found when brought in to do performance tuning: poorly factored code, no repeatable tests, and a management that had lost confidence in the project. He went to Chrysler Information Services management and told them what he had found, and that he knew how to fix it. Throw all the existing code away! The first full XP project was born.

We brought Kent in as head coach; he would spend about a week per month with us. Ron Jeffries was brought in as

Kent's full-time eyes and ears. The fixed-price contract was cancelled, and about one-half of the Chrysler developers were reassigned. Martin Fowler, who had been advising the Chrysler side of the project all along and clashing with the fixed-price contractor, came in to help the customers develop user stories. From there, we followed Kent as he made up the rules of XP. A commitment schedule was developed, iterations were laid out, rules for testing were established, and paired programming was tried and accepted as the standard.

At the end of 33 weeks, we had a system that was ready to begin performance tuning and parallel testing. Ready to begin tuning because it was well factored and backed up by a full battery of unit tests. And, ready to begin parallel testing because a suite of functional tests had shown the customers that the required functionality was present.

That increment of C3 launched in May 1997, not as soon as we had hoped. We were slowed by two factors. First, we had decided to replace only the internals of the payroll system. We left all of the external interfaces intact. Matching up the output from our new system to the old payroll master ended up being a much larger task than we had originally estimated. Second, we decided not to launch during any pay period with special processing require-



Team: 10 programmers, 15 total

Application: large-scale payroll system

Time: four years

ments, such as W-2 processing, profit sharing, or general merit pay increases. This effectively eliminates November through April.

Since the launch of the monthly system, we've added several new features, and we have enhanced the system to pay the biweekly paid population. We have been paying a pilot group since August 1998 and will roll out the rest before the Y2K code freeze in November 1999.

Looking back on this long development experience, I can say that when we have fallen short of keeping our promises to our management and our customers, it has been because we have strayed from the principles of XP. When we have driven our development with tests, when we have written code in pairs, when we have done the simplest thing that could possibly work, we have been the best software development team on the face of the earth.

Underestimation

From time to time you will commit to more than you can accomplish. You must reduce the occurrence of underestimation as much as possible by getting lots of practice estimating. If you are overcommitted, you first try to solve the problem in the family. Have you slipped away from the practices? Are you testing, pairing, refactoring, and integrating as well as you can? Are you delivering more than the customer needs in places?

If you can't find any way to go faster, you have to ask the customer for relief. Staying committed to more work than you can confidently complete is a recipe for frustration, slipping quality, and burnout. Don't do that. Re-estimate based on what you've learned, then ask the customer to reschedule. We can only complete two out of three stories, so which two should we finish and which one goes in the next iteration or release? Is there a story that has more critical parts and less

critical parts so we can split it and deliver the most important parts now and the less important parts later?

Uncooperative customers

What if you get a customer who just won't play the game? They won't specify tests, they won't decide on priorities, they won't write stories. First, by completing functionality iteration after iteration, and by giving the customer clear control over development, you are trying to build a trust relationship with the customer. If trust begins to break down, figure out if it's your fault. Can you do a better job of communicating?

If you can't solve the problem on your own, you have to ask the customer for help. Extreme programmers simply don't go ahead based on their own guesses. Explain or demonstrate the consequences to the customer. If they don't change, make your concerns more visible. If no one cares enough to solve the problem, perhaps the project isn't a high enough priority to go on.

Turnover

What if someone leaves? Won't you be stuck without documents and reviews? First, a certain amount of turnover is good for the team and for the people on the team. However, you'd like people to leave for positive reasons. If programmers go home at the end of every week seeing the concrete things they have accomplished for the customer, they are less likely to get frustrated and leave.

When someone leaves an XP project, it isn't like they can take away any secrets that only they know. Two people were watching every line go into the system. And whatever information does walk out the door, it can't hurt the team too much because they can run the tests to ensure that they haven't broken anything out of ignorance.

New people on an XP team spend the first couple of iterations just pairing with more experienced people, reading tests, and talking with the customer. When they feel ready, they can accept responsibility for tasks. Over the course of the next few iterations, their personal velocity will rise as they demonstrate that they can deliver their tasks on time. After a few months, they are indistinguishable from the old salts.

Programmers that don't work out with the team are a problem, too. XP is an intensely social activity, and not everyone can learn it. It also requires aban-

doning old habits, which can be difficult, especially for high-status programmers. In the end, though, the many forms of feedback in XP make it clear who is working out and who isn't. Someone who consistently doesn't complete tasks, whose integrations cause problems for other people, who doesn't refactor, pair, or test Everyone on the team knows the score. And the team is better off without that person, no matter how skilled.

Changing requirements

The bugaboo of most software development is just not a problem in XP. By designing for today, an XP system is equally prepared to go any direction tomorrow. Things that are like what you've already done will be easier, just by the nature of refactoring to satisfy "once and only once," but those are precisely the things that are most likely to happen. However, should a radically new requirement arise, you won't have to unwind (or live with) a lot of mechanism built on speculation.

I didn't initially realize the extent to which XP can adapt to changing requirements. The first version of XP assigned stories to all the iterations in a release, as part of release planning. The team discovered that they could get better results with less planning by only asking the customer to pick which stories should be in the present iteration. If a new story comes up, you

Ford Motor: A Unique Combination of Agility and Quality

Don Wells, Ford Motor

Finance Systems at Ford Motor has been developing the Vehicle Costing and Profit System (VCAPS), an analysis tool that produces reports on production revenues, expenses, net income, and profit. The input is a bill of materials, fixed costs and expenses, and variable costs such as labor hours. VCAPS assembles this data into detailed cost analysis reports to support corporate-level forecasting and decision making.

Ford started VCAPS in 1993 and built it with VisualWorks and GemStone Smalltalk. VCAPS is now being maintained with a small staff and is to be replaced with a newer system.

The VCAPS project challenged us two ways. First, the analysts wanted modifications and new functionality before each run. Constantly changing requirements kept us in reaction mode. We never caught

up. Second, the system needed to be run in a limited span of time. But the system took a long time to process and required lengthy manual input before producing final output. A bug could waste precious time by requiring a rerun.

XP offered us a unique combination: agility to meet the volatile requirements on time and quality to avoid the dreaded rerun.

We began XP with the planning game. It was a failure. Customers and management were unaccustomed to negotiating schedules. The commitment schedule produced was perceived as lacking credibility and utility. We had to swap in Microsoft Project schedules, which could be modified without large meetings and could produce the kinds of artifacts management was used to seeing and taking action on.

We continued by adding a few unit tests. Automated unit testing was an enormous success. After a year, we had 40 percent test coverage and management had measured a 40 percent drop in bug reports. XP was being noticed.



Team: 12 programmers, 17 total
Application: cost analysis system
Time: six years

We solved problems by adding XP practices. Tests enabled continuous integration and small releases. These allowed us to roll in collective ownership and refactoring. We were working toward simple design. Building momentum, we tried pair programming. We had to work hard to get pair programming going. Our developers found it awkward; it took a while to become comfortable.

After a year and a half, the decrease in system failures had reduced the number of emergency releases to a point where customers and managers noticed far greater system stability. Overall, XP was very successful in our environment.

Tariff System: Tests You Can Read

Rob Mee, Independent consultant

Tariff System is a subsystem of a large Smalltalk/GemStone project at a major international container-shipping company. Using XP practices, Tariff System was taken from inception to production in three months by a team of three. The resulting system proved to be unusually stable and easy to maintain.

At the outset of the project, the team resolved to adhere to several core XP practices: always program in pairs, use the simplest design possible, refactor aggressively, and write extensive unit tests. All of these practices were very effective. One XP idea that initially seemed far-fetched was writing tests before writing the code that satisfied them. We were surprised to find that in fact this helped bring our designs into focus, enabling us to work more quickly.

Another practice we employed from the beginning was collecting requirements from users in the form of user stories. We

had mixed results with this. As programmers focused on coding, we found the role of facilitating and negotiating with users difficult. More important was the fact that users needed lots of help writing stories that were both relevant and unambiguous. In the end, we felt that perhaps XP was missing a project role. We needed someone from the development team whose primary focus—and particular talent—was interacting with users.

In our efforts to refactor test cases and fixtures, we discovered that creating little languages for our major domain objects dramatically improved the readability and brevity of our test code. It also practically eliminated the time we spent thinking about how to create object instances when writing tests. We defined grammars for about ten of our domain classes. Here's a simple example used to construct a Service Offering:

```
newFromString: 'from Oakland to
Tokyo shipping toys: 20ft containers $500;
40ft containers $1000'.
```



Team: three developers

Application: shipping tariff calculation system

Time: three months

The constructor uses a parser, automatically generated from a grammar, to produce the domain object. The code to instantiate this object using standard constructors would have taken many lines, would have been difficult to read, and would have distracted from the test case itself.

Eventually, we discovered that we could combine the individual domain languages into a larger description of the system as a whole, which proved to be a valuable tool in the expression of functional tests.

don't have to shuffle the remainder of the iterations, you just put it in the pile. One or two weeks later, if the story still seems urgent, the customer will pick it.

Planning one iteration at a time also introduces a pleasing self-similarity. At the scale of months and years, you have the stories in this release and then the stories in future releases. At the scale of weeks and months, you have stories in this iteration and then the stories remaining in this release. At the scale of days and weeks, you have the task you are working on now and then the rest of the tasks in the iteration. And at the scale of minutes and days, you have the test case you are working on now and then the rest of the test cases that you can imagine.

XP is by no means a finished, polished idea. The limits of its application are not clear. To try it today would require courage, flexibility, and a willingness to abandon the project should your use of XP be failing.

My strategy is first to try XP where it is clearly applicable: outsourced or in-house development of small- to medium-sized systems where requirements are vague and likely to change. When we begin to refine XP, we can begin to try to reduce the cost of change in more challenging environments.

If you want to try XP, for goodness sake don't try to swallow it all at once. Pick the worst problem in your current process and try solving it the XP way. When it isn't your worst problem any more, rinse and repeat. As you go along, if you find that any of your old practices aren't helping, stop doing them.

This adoption process gives you a chance to build your own development style—which you will have to do in any case—to mitigate the risks should XP not work for you and to continue delivering as you change. ♦

Reference

1. S. Prowell et al., *Cleanroom Software Engineering*, Addison Wesley Longman, Reading, Mass., 1999.

Kent Beck owns and operates First Class Software, your typical one-person consulting company masquerading behind a fancy name and an answering machine. In addition to two books and 50 articles, he is the author of the forthcoming Extreme Programming Explained: Embrace Change (Addison Wesley Longman, Reading, Mass., 2000). Contact him at kentbeck@csi.com.

Hitting the Target: Adding Interaction Design to Agile Software Development

Jeff Patton
Development Team Leader
Tomax Technologies
224 South 200 West
Salt Lake City, UT 84101
USA
1.801.924.6924
jpatton@tomax.com

ABSTRACT

Extreme Programming appears to be a solution for discovering and meeting requirements faster (through close customer collaboration) as well as creating quality software. In practice we found XP did deliver high quality software quickly, but the resulting product still failed to delight the customer. Although the finished product should have been an exact fit, the actual end-user still ended up slogging through the system to accomplish necessary day-to-day work. This paper describes using interaction design in an agile development process to resolve this issue. Using interaction design as a day-to-day practice throughout an iterative development process helps our team at Tomax Technologies deliver high quality software, while feeling confident the resulting software will more likely meet end-user expectations. The method of Interaction Design followed here is based on Constantine and Lockwood's Usage-Centered Design. Recommendations are provided on how to practice an agile form of U-CD and how to incorporate bits of Interaction Design thinking into every day development and product planning decisions.

Keywords

Agile Methodologies, Interaction Design, Usage-Centered Design, Extreme Programming, Requirements Gathering.

1. INTRODUCTION

This experience report discusses my discovery and incorporation of Constantine & Lockwood's Usage-Centered Design [6] into the day to day work my team does to deliver high quality software.

Summarizing observations of several projects in Reflective Systems Development [13], Mathaisen observed "Systems development methods were seldom, or only partially, followed by experienced practitioners." We were no exception. Our current form of U-CD uses new skills taught by Larry Constantine & Lucy Lockwood. In addition we've made several modifications to the process to accommodate time limitations, information limitations, and an iterative development environment. The result is a situation specific "agile" form of U-CD that fits tightly into our team's local methodology.

When incorporated into an agile development process, the interaction design concepts in Usage-Centered Design pack a powerful 1-2 punch: Agile development methods allowed us to

deliver high quality software sooner, and interaction design concepts lent us the degree of end-user empathy we were missing to help increase confidence that we hit our target of end-user satisfaction. All this results in our team being more successful today than 2 years earlier.

2. IDENTIFYING THE PROBLEM

2.1 There Has to Be Better Way.

I've spent years developing software "traditionally." Basically, this consisted of a blend of Waterfall Methodology and complete chaos. I saw intelligent folks work very hard to identify requirements, create a thorough definition of scope and functional design, approve that, and then finally build it. More often than not the resulting software would miss its target. It was often late. There were problems with quality – the software released with bugs. But even with quality issues resolved, the resulting software was hard to understand and cumbersome to use. It didn't seem to be appropriate for the actual work the end-users were trying to accomplish. Requirements were often missed in the design phase, resulting in features necessary to automate the business process being left out of the software. Features originally thought important during the design phase were often discovered to be unnecessary and went un-used.

Watching this cycle over and over left coworkers and customers paralyzed with fear. We analyzed and designed longer with the hope we'd get it right the next time. Customers reviewed designs longer or delayed reviewing them out of fear they'd miss something and be blamed for inevitable omissions in the delivered product. We started developing late. We finished even later.

There had to be a better way.

3. FINDING THE SOLUTION

3.1 Enter Extreme Programming

Extreme Programming [3] surfaced as an alternative to this madness along with other new ways of developing software – now branded as Agile [1]. Surely close customer collaboration and iterative development would correct customer satisfaction issues. Surely test-driven programming [16], pair programming, and aggressive refactoring [11] would improve software quality.

I stumbled into and spent a valuable year with Event Solutions, a company committed to XP principles. We built high quality

software at an aggressive rate. Deliveries were on time and with the expected scope usually intact. However, I still found the company missing targets. The resulting product seemed to have features the actual end user didn't need or care about while lacking features the end user did need. Much time seemed to be spent on features that would go un-used by actual end users. These same actual end users had to devise lengthy procedures to force their actual business processes to work with the software that was built. Shouldn't close customer collaboration have mitigated this issue?

Ideally an XP customer is an expert end-user employed by the company actually purchasing the software. In Evant's case they were indeed expert users at one time, but now as product managers working for Evant, they had the responsibility to deliver commercially viable software to be competitive with other products in the same marketplace. They had to balance the needs of users we currently had with users we hoped to acquire. This was a daunting responsibility requiring tough trade-offs.

At delivery time, it wasn't always clear where those tradeoffs had been made. Product managers seemed to be surprised that actual end-user needs had not been met. We moved into a reactive mode delivering software that we hoped would address end-user needs, then waiting for the inevitable requests to make changes. An element of empathy with the actual end-users seemed to be missing. We were guessing what they needed. Was this just an unavoidable challenge of software development?

3.2 Beauty Is Only Skin Deep.

At Evant we'd always worked hard to make our software look good and be easy to use. Our UI specialist did a fabulous job with screen design and the product was consistent and easy to understand. However we still had the issue with actual business processes being hard to accomplish in the software and important parts of business processes being left out completely. Sure it looked good, but apparently there was more to hitting the target than looks.

3.3 We're All Crazy.

Years ago I'd read Alan Cooper's About Face [9]. It contained lots of good information on what not to do when designing the software's user interface. But our software seemed to have a good user interface – at least it didn't break any of the major rules.

During the spring of 2001 I was able to hear Cooper speak in Berkeley. His focus was less on bad screen designs and more on software missing its target as a result of not understanding its user. He pointed out the necessity of a *persona*. A *persona* was a walking, talking, fictitious user with well-developed fictitious needs and concerns. Reading Cooper's The Inmates are Running the Asylum [10], I found that as a technologist, I'd likely never be able to identify with my user. I, and the folks I worked with were the inmates and it would take serious effort to think like the *persona* we could create. Knowing you have a problem is half the battle. I proceeded under the assumption I could find the steps to recovery.

In an attempt to move forward, I latched onto what looked like a valuable starting point. Alan Cooper used the term "interaction design" to describe the missing role in software development, and further said "Almost all interaction design refers to the

selection of behavior, function and information and their presentation to users." [10] Looking back at challenges I'd experienced before it seemed that we did a poor job of selecting the appropriate behavior and functions to implement in the software.

4. FINDING THE SOLUTION, AGAIN

4.1 It's not Chet's Fault.

While lurking in the ExtremeProgramming discussion group [18], I read Ron Jeffries recommendation of Constantine & Lockwood's Software For Use [6] as a possible source for good information on user interface design. Although Extreme Programming Installed [12] may encourage blaming Chet – I'll assign Mr. Jeffries the responsibility for starting me down this path.

Like Cooper's concerns, Constantine and Lockwood's justifications for effective user interface design and usability were preaching to the choir. But something different in this book was an actual documented method for arriving at a usable piece of software including choosing appropriate behavior and functions for the people identified as users of the software. However, the process described looked complicated, time consuming, and not easily adapted to an agile development approach. What's more it needed to happen up front. Adding a time consuming process to the front end of software development sounded too much like the bad experience I had been running from.

4.2 Ah-Ha!

During the summer of 2001, I had the opportunity to learn Usage-Centered Design from Larry & Lucy directly. The book is thick – and I'd wondered how we were going to compress this process into a weeklong class. As exercises between lectures we discussed a business problem, brainstormed ideas onto 3 x 5 cards and saw models emerge almost magically by shuffling cards around the table. We did this collaboratively in a group with lots of discussion. We learned an effective way to move from these arranged models of cards on the table to wireframe user interface. We learned how to validate – or test – our user interface using the information we'd put together still on those 3 x 5 cards.

The business problem we took on to solve in exercises seemed daunting at first. But surprisingly, in a short amount of time we arrived at an effective design. And what's more, the whole process was understandable and fun. If this was Usage-Centered Design in practice, I could easily see it used as a collaborative approach to generating story cards for use in Extreme Programming development. This would surely result in us delivering software that was high quality and effective at meeting the real business needs of the user. Simultaneously I stumbled onto an assertion on page 122 of Cockburn's Agile Software Development [5] that when we look at the scope of concern for Usage-Centered Design and XP that the two sets of practices could indeed inhabit the same project.

If past problems sprung from sometimes selecting and building the wrong behavior, possibly using U-CD as a method for interaction design would result in selecting correct behavior more often. With XP we could now accurately plan, develop, and release a set of features. We could hit the target of on-time delivery and high quality. With more method behind choosing

the features to implement, we now hopefully had a target that more likely included end-user satisfaction.

5. DEFINING THE SOLUTION

5.1 Agile Usage Centered Design

Although Usage-Centered Design is thoroughly explained in Software for Use [6], an Agile approach is first documented in Larry Constantine's paper: "Process Agility and Software Usability: Toward Lightweight UsageCentered Design" [8]. The steps given here are an abbreviated overview of the process. This is Constantine and Lockwood's process with a few minor variations to match the way my team and I practice it today.

1. Identify participants.

Sequester a diverse mix of people in a room to collaborate on this design. Include domain experts, business people, programmers and test/QA staff. Include a facilitator that knows this process.



Figure 1. Collaborative design sessions include a diverse mix of people.

2. Preconception purge.

Let loose. Everyone brain-dump about the software we need to write. Complain about the product you're replacing. Explain the cool features you expect the new product to have. Get everyone's concerns out into the open. Write these concerns down in plain sight on whiteboards or poster sized paper hung on the wall.

3. Review the domain.

Domain experts and users in the room explain the business process, as it exists today. Who is involved in the process? What combination of manual processes and computer based tools do current participants engage in to meet their goals?

4. Define user roles and role model.

Brainstorm user roles onto 3 x 5 cards. Who will be using this software? What are their goals? Prioritize the roles by shuffling the stack of cards. Note the most important roles. Label those roles *focal* roles. Place them in an arrangement

on the table that makes sense with similar roles closer to each other. Discuss the relationships these roles have with each other. This is a role model.



Figure 2. Fixing role cards to poster paper and annotating relationships allows the role model to be posted for everyday reference.

5. Define tasks and task model.

Now that we know who will use our software, brainstorm tasks these roles will be doing to accomplish their goals onto 3 x 5 cards. Shuffle the cards to prioritize them based on importance, then on frequency. Note the most important and most frequent. Label those tasks *focal* tasks. Arrange the cards on the table. Place tasks similar to each other, or dependent on each other, together. Place tasks that have nothing to do with each other further apart. Discuss the relationships these tasks have with each other. This is a task model.

6. Define interaction contexts.

You'll find tasks in the arrangement on the table clump up. Grab a clump. This is an interaction context. Give the interaction context an appropriate name.

7. Detail user tasks.

For each task in your interaction context, write a Task Case directly on the card. The Task Case takes the form of a conversational Use Case similar to that described by Rebecca Wirfs-Brock in [17]. Alistair Cockburn in Writing Effective Use Cases [4] might classify them as "system scope, sea-level goal, intention-based, single scenario, Wirfs-Brock use case conversation style." U-CD would encourage you to simplify and generalize these Task Cases. Using a conversational form makes them easy to read. Limiting the scope and goal keeps them from being too broad or too detailed. Generalizing them keeps them short and allows deferring user interface details for implementation time.

8. Create an Abstract Prototype.

For each interaction context, using the task cases you've detailed create an abstract user interface prototype. This process is best described in [7]. At the end of this process

you'll know what components will be on the interaction context.

9. Create wireframe user interface.

Using pencil and paper create a wireframe drawing of the interaction context. Show basic size and placement of screen components.

10. Test the interaction contexts.

Use role-playing to step through each task case used in the interaction context. One participant pretends to be the role that would perform the task, another plays the role of the user-interface. Validate that you can easily and effectively reach your goal using this interaction context.

6. PUTTING IT INTO PRACTICE

6.1 Starting In the Middle.

Armed with a year's worth of Extreme Programming development experience, U-CD training, and lots of other bits of useful information from books, papers and colleagues, I set out at Tomax, my current employer, to prove that U-CD + XP was indeed a potent combination that would lead to on time delivery, high quality and ultimately satisfied users. The rest of this paper describes how close we came and how much remains to be discovered.

While it's exciting to think we could put into place a set of new practices, we never quite have a clean slate. In my situation we had legacy practices to deal with. When it came time to apply Usage-Centered Design it was often a bit too late. There was no shortage of new software to write, but before our company had agreed to write the software, documents had generally been written up and agreed-to describing scope, features, and functionality. In many cases if we were to attempt to practice U-CD our company would have been accused of re-trenching the same material already discussed by marketing and/or project management. Looking at the use of the software often meant asking users to repeat conversations they'd already had when drawing up the agreement. In addition the results of such a conversation may yield changes in scope. This notion was at best unpopular.

6.2 Some Opportunities and Some Success

There were, however, some greenfield opportunities. These were projects where requirement were not yet agreed to and where the customers and management were willing to approach things in a slightly different way. In those situations we practiced Agile U-CD as described above with some success.

6.3 What Worked:

The preconception purge before the process seemed to be the chance to vent everyone was looking for. Giving the group permission to have an unorganized conversation where anything could be said brought to light many concerns and fears we'd have not gotten to any other way. This free form conversation supplied everyone involved with an immense amount of useful background. We left ideas captured during this process on poster paper taped to the wall. At the end of this process we were able to double back and make sure we'd dealt with the concerns, or found they weren't really concerns any more.

Working with 3 x 5 cards struck some participants as very low tech, but the results were very effective. The discussion took the same form as a CRC card session [2] might take. But, instead of classes, responsibilities, and collaborations, we talked about user roles and tasks. We saw lots of card waving and passing cards back and forth. People immediately understood what was important by looking at the position of the card on the table. People immediately knew what ideas were related by their position in relation to each other. An arrangement of cards on the table could communicate far more, faster than any paper document or diagram could. We found that taping card arrangements to poster paper, then marking up the taped arrangements resulted in a very valuable model.



Figure 3. Participants quickly learn to work with 3 x 5 cards.

Mapping Task Cases to Abstract Prototypes was a very simple and effective way to push through from knowing what we needed to do to how it might look on the screen. The Abstract Prototype consisted of post-it notes, signifying abstract components, stuck to poster paper. We could easily rearrange them and push through this paper-prototyping phase to a simple wireframe user interface.

6.4 What Was Bumpy:

Folks had problems with User Roles. In U-CD a role isn't a job title – but more accurately a high level goal. For example: Clerk is a job title. CustomerSalesTransactionHandler is a role. The distinction becomes important when someone looks at a list of roles later and is unable to determine what each does. Or when looking at a task case like ReturnMerchandise and ask who does it? In this case if you're using job titles, the Clerk, Assistant Manager and Manager may all have responsibility to perform that task – but, we'd have to know the business rules to be sure. However, we can reasonably assume a CustomerSalesTransactionHandler might have that responsibility. Choosing expressive role names is valuable – but is a hard idea to grab onto for domain experts. In practice I found it easier to let folks use roles like "clerk" initially. During discussion of the role and the goals the role had, we could easily

convert the job title to one or more role names that captured the users' goals.

Attention spans weren't long enough. By the time you reach the tail end of the process when it will really bear fruit, people are exhausted and unable to effectively do a good job building the UI. Reconvening the next day left us with a fair amount of ground to cover again to get everyone back on the same page. The process takes a while and for those who don't do it often, it's time consuming and tiring. Folks were accustomed to one person going off to a cubical to write functional specifications and not this long collaborative process. As anyone who practices pair programming can tell you, constant collaboration can be exhausting. We found it most effective to split the process at the point we'd identified interaction contexts. We could then continue the process at a later time using a smaller more focused group of people – those that were ultimately responsible for delivering the system.

The resulting artifacts look funny. In this organization functional design previously took the form of a list of "shall"s—the software "shall do this" sort of statements along with assumptions, a very literal screen design, and sometimes a narrative on how it would be used. Roles and a role model weren't immediately understandable. Task Cases seemed too general – too abstract for some folks. Wireframe UI drawings weren't quite literal enough. These issues impacted acceptance of the functional design. On occasions that we needed to produce functional design, it seemed to work best to document user roles, the names and goals of each user task, and cleaned up versions of wireframe user-interface drawings. These things dropped into a document seemed to look enough like requirements for folks to "sign-off" on the effort.

7. REFLECTING ON WHAT WE'D DISCOVERED

7.1 Were We Gaining Anything?

It sure felt that way. Although close collaboration within a large group was tiring, when we finished the amount of tacit knowledge in the group was irreplaceable. Everyone within the team understood who the users were and what their goals were. Those in the team who hadn't been present for the U-CD sessions quickly assimilated the vocabulary of those who did. Artifacts, such as role and task models, created during the session were posted in the development area to "radiate" [5] information throughout the implementation of the software.

Our priorities became clear. We need only find the focal, or most important user roles and their focal task cases to find the best starting point for development. If we became bogged down implementing functionality for roles that weren't focal, we could justify choosing a simpler, less elegant, but cheaper and faster approach.

Was this better than a long, functional design written by one expert? It's not easy to say that the results were definitely better, but it is easy to say that team members' understanding and ownership of the software was higher than before. By arriving at this functional design together, all knew how to accomplish this process and we'd eliminated what was before a single point of failure. This seemed like a definite improvement.

7.2 Test-Driven Design For User Interactions

Throughout the development process, whenever anyone on the team was unclear on the direction we were going with the software, we'd pick up the original task-cases and attempt to execute them on the software. They became our working acceptance tests.

Knowing user roles helped answer other questions – like what the ability level of the user was and what that user's goal was. For example, often in a business process the goal of the user doing the process is much different than a manager who needs to have visibility of what was done. They need to see different information at different times. Using user roles, circumstances like this became clearer.

Finally, when formal acceptance and QA had to occur, task cases could be "fleshed out" to contain specific references to the actual implemented user interface along with literal test data. Roles would serve as a collection point for acceptance tests. We'd focus on validating the software a role at a time essentially wearing the hat of the user role and performing the work they'd need to perform with the software.

Our confidence in the finished software was higher. The feeling seemed analogous to the feeling you get developing source code using automated unit testing and test-driven development. It's not really provable that code developed this way is better than other ways, but after doing it I find my confidence in the code is higher. I also find I'm unwilling to work any other way as that seems risky or foolish. As with test-driven development, there was no knowing if our finished results were indeed better than we could have come up without UCD, but confidence was higher. Proceeding on a project without knowing what user roles existed for the product and what tasks they needed to perform now feels as risky as writing code without unit tests.

8. WHAT SHOULD I DO TOMORROW?

8.1 Interaction Design Incorporated Into Day-to-Day Processes of a Mostly Agile Company.

At Tomax Technologies, certain agile processes have taken off and work well. Scrum-style [15] daily stand-up meetings are commonplace. Cockburn's Information Radiators abound [5]. Teams develop iteratively, many of them using schedules generated by an XP style planning game. Some teams religiously use unit testing, pairing, and refactoring. Other teams are still a bit suspicious of all these newfangled ideas. Although we have product managers, they don't have the time to ride shotgun on a project the way an XP customer should. They rely on the team to make the detailed decisions about the implementation of features in the product. Acceptance testing is up to the team and performed by test/QA staff assigned to the team. Development methodology is a decision made more at the team level than the corporate level. In this sort of environment, how do we incorporate some interaction design into things we do every day?



Figure 4. Our development environment at Tomax is wallpapered with role models, task models, task cases, and wireframe UI drawings alongside XPstyle iteration schedules.

The following is a short list of InteractionDesign-centric guidelines our team tries to observe:

1. We always ask “who?”

While we're looking at a piece of development we make an effort to understand who will be using it. What is the user role involved? If we don't know, we back up and do a quick user-role brainstorming session. Arrange a few 3 x 5 cards on a table to understand the role model, and then continue on. When we understand who will be using the application, we make better decisions about what they should see and how sophisticated the interactions can be.

2. We validate user interactions with a taskcase.

To make sure our user interface is usable, we write a simple task-case giving us the step-by-step intention driven process a particular user role might follow to complete the task. Does the current design of the application do this efficiently? This may be analogous to a manually executed XP acceptance test.

3. We strive to understand focal user roles and focal task-cases.

Make sure everyone in the project understands who it is most important to satisfy and what specific activities need to run smoothest. Focus on those. Spend extra effort to make them right. Allow the less important roles and task-cases to slide. They need to be functional - but fluid and pretty may be a little less important. Time is most wisely spent elsewhere.

4. We look for features that don't serve any role or facilitate any task.

There's always a temptation to scoop up seemingly easy features. Beware statements like "It would be cool if the software could..." - or - "right here we could show..." Always ask what user role needs this? What will they be doing when they do need it? Does this user role care about this information? What information do they care about?

5. We elevate the writing of stories into interaction design.

Help the folks who know the business understand user roles and task-cases. Before requirements are created discuss roles - who's important, who isn't. Discuss task cases – what does each role do. Clearly understand priority and dependence. This makes planning an iteration easier. This allows us to deliver a truly usable product sooner by appropriately accommodating all the necessary tasks of a focal role.

6. We revisit our requirements often.

In implementing the software thus far, have we learned of a role we didn't know about earlier? Have we found that to accomplish a goal it may take unforeseen tasks or that some of our tasks are unnecessary? When we're not sure, we pull out the 3 x 5 cards and reassemble role models and task models to evaluate if the design still makes sense.

9. INTERACTION DESIGN AND AIM

9.1 Beck & Cooper Face Off.

In an interview posted Jan 15th 2001 on Fawcette Technical Publications website [14], Kent Beck and Alan Cooper face off on the subject of up-front interaction design vs. agile methods. Excerpts from the conversation include the following comments.

Cooper: "...I'm not talking about having a more robust communication between two constituencies who are not addressing the appropriate problem. I'm talking about incorporating a new constituency that focuses exclusively on the behavioral issues. And the behavioral issues need to be addressed before construction begins."

Beck: "OK, wait. I agreed with you very close to 100 percent, then you just stepped off the rails. I don't see why this new specialist has to do his or her job before construction begins?"

Cooper: "It has to happen first because programming is so hellishly expensive... There's enormous cost in writing code, but the real cost in writing code is that code never dies. If you can think this stuff through before you start pouring the concrete of code, you get significantly better results."

Beck: "No. I'm going to be the programming fairy for you, Alan. I'm going to give you a process where programming doesn't hurt like that—where, in fact, it gives you information; it can make your job better, and it doesn't hurt like that. Now, is it still true that you need to do all of your work before you start?"

I hear Cooper asserting that software is too rigid to easily change – that we must get interaction design right, all of it, before we develop. I hear Beck saying that we've eliminated the cost of change curve so we can now get it wrong without incurring great expense. It seems that both Beck and Cooper share the same goal of cost-effectively delivering high quality software that results in end-user satisfaction. They seem to disagree on how this is done. Could they both be right to some degree?

9.2 Building Better Aim.

If our goal is to deliver high quality software on time while satisfying end-users, then, that's a good target to aim for. I'd interpret Cooper as saying we need to hit our target with one carefully calculated shot, and the interaction designer should be the one to take aim. I'd interpreting Beck saying we can shoot

often and cheaply, so keep shooting until you hit your target. Let businesspeople take aim since they're paying for all of this.

In my experience, I've seen evidence that we can indeed shoot often and cheaply. But, I've also seen evidence that businesspeople don't always have the best aim. And, although XP and agile methods do help minimize the cost of developing working software and decrease the cost of changing it, cost is still cost. And businesspeople don't like paying unnecessary costs.

If this metaphor holds, then a working solution might be to try to improve the aim of the businesspeople by using interaction design concepts to help better define our requirements. If we can dependably and repeatably apply interaction design tactics we should be able to build better aim.

Our experience at Tomax bears this out. The simplicity and repeatability of U-CD allows the actual customer, business leaders, and developers to all participate in "designing" the requirements. During this process we all feel more confident that we understand what the software should do and why. We still miss our target sometimes, however good development practices do indeed allow us to change the design quickly. Also important is that when we do get it wrong we now understand a little better why. It's often an undiscovered user role, or goal. Using an interaction designer's sensibilities and UCD as process framework, we are all learning to ask better questions – which gives us the better aim we've been looking for

10. ACKNOWLEDGEMENTS

Thanks to valued team-members from Tomax Technologies & Evant Solutions for providing a laboratory to learn in. Thanks to Larry Constantine & Lucy Lockwood for being great teachers. Thanks to collaborators and advisors: Stacy Patton and Kay Johansen. Special thanks for valuable feedback and advice goes to Alistair Cockburn for help in motivating and revising this paper. Thanks to Lougie Anderson for her advice, and encouragement. Thanks also to the enthusiastic team at Sabrix who allowed themselves to be guinea pigs.

11. REFERENCES

- [1] Agile Alliance <http://www.agilealliance.com>

- [2] Beck, K., Cunningham, R., A Laboratory For Teaching Object Oriented Thinking, (1989) <http://c2.com/doc/oopsla89/paper.html>
- [3] Beck, K., Extreme Programming Explained, Addison-Wesley (1999)
- [4] Cockburn, A., Writing Effective Use Cases, Addison-Wesley (2000)
- [5] Cockburn, A., Agile Software Development, Addison-Wesley (2001)
- [6] Constantine L. & Lockwood L., Software For Use, Addison - Wesley, (April 1999)
- [7] Constantine, L., Windl, H., Noble, J., & Lockwood, L. From Abstract to Realization in User Interface Designs: Abstract Prototypes Based on Canonical Abstract Components (2000) <http://www.foruse.com/Files/Papers/canonical.pdf>
- [8] Constantine L., Process Agility and Software Usability (2001) <http://www.foruse.com/Files/Papers/agiledesign.pdf>
- [9] Cooper, A., About Face, Hungry Minds Inc. (1995)
- [10] Cooper, A., The Inmates are Running the Asylum, Sams (1999)
- [11] Fowler, M., Refactoring: Improving the Design of Existing Code, Addison -Wesley (1999)
- [12] Jeffries, R., Anderson, A., Hendrickson, C., Extreme Programming Installed, Addison-Wesley (2000)
- [13] Mathaisen
- [14] Nelson, E., Extreme Programming vs. Interaction Design (2002)http://www.fawcette.com/interviews/beck_cooper/
- [15] Schwaber, K., Beedle M., Agile Software Development with Scrum, Prentice Hall, (2001)
- [16] Test Driven Programming <http://xp.c2.com/TestDrivenProgramming.html>
- [17] Wirfs-Brock, <Which paper did she first document conversational use cases? Find this.>
- [18] Extreme Programming Yahoo Group <http://groups.yahoo.com/group/extremeprogramming/>

Hitting the Target: Adding Interaction Design to Agile Software Development

- Jeff Patton
- Tomax Technologies
- Salt Lake City, UT
- Jpatton@tomax.com



In a Nutshell

- **Life in a chaotic waterfall style development environment was tough.**
 - Quality was poor.
 - Projects were late.
 - End-users were dissatisfied.
- **Life in an XP world was immensely better.**
 - Quality was good.
 - Projects were on time.
 - But, actual end-users were still not thrilled.
- **Usage-Centered Design, in its Agile form, complemented an XP style development process.**
 - Quality was good.
 - Projects were on time.
 - Actual end-users seemed happier.



2

Usage-Centered Design

- U-CD is an instance of Interaction Design published by Constantine & Lockwood in Software For Use.

Interaction Design:

“Almost all interaction design refers to the **selection** of [software] behavior, function, and information and their **presentation** to users.”

Alan Cooper, The Inmates are Running the Asylum

- Interaction Design concepts help us **select behavior & function.**



Collaborative U-CD Sessions

■ Involve:

- Developers
- Interaction Designers
- Domain Experts
- Business & Marketing
- Actual End-Users



Collaborative U-CD

TOMAX®
retail-in-realtime™



The Process In a Nutshell

■ Use low-tech tools:

- Poster paper
- 3 x 5 cards
- Post-it notes
- Markers
- Tape
- Food



■ Discuss any preconceptions.

- (Preconception Purge)

TOMAX®
Discuss the domain.
retail-in-realtime™

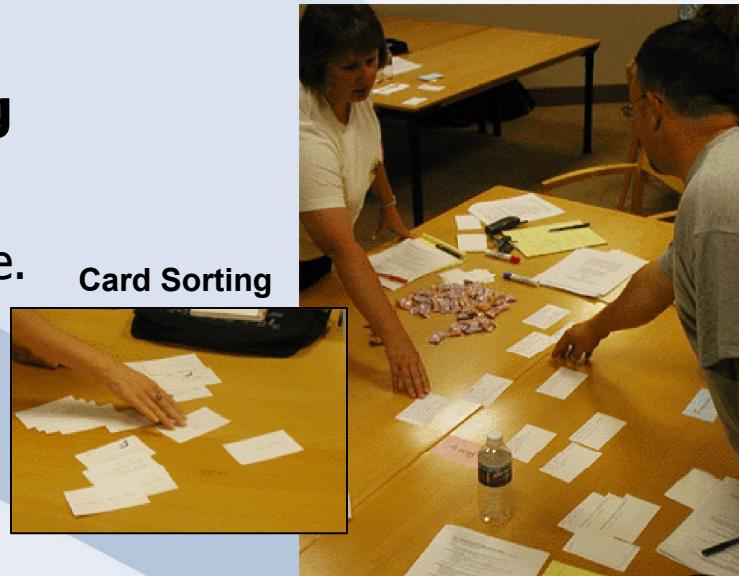
■ Brainstorm user roles onto 3 x 5 cards.



The Process: 3 x 5 Card Games

- **Model the roles using card sorting & arranging techniques.**

- Identify the primary goals of each role.
- Identify **focal** roles.
- Identify relationships between roles.



- **Brainstorm the tasks these roles perform to meet their goals.**

- **Model tasks using card sorting & arranging techniques.**

- Identify goals for each task.
- Identify which roles perform each task.
- Identify ~~frequency and distribution~~ ^{task and triage tape} then determine **focal** tasks.



TOMAX
retail technology

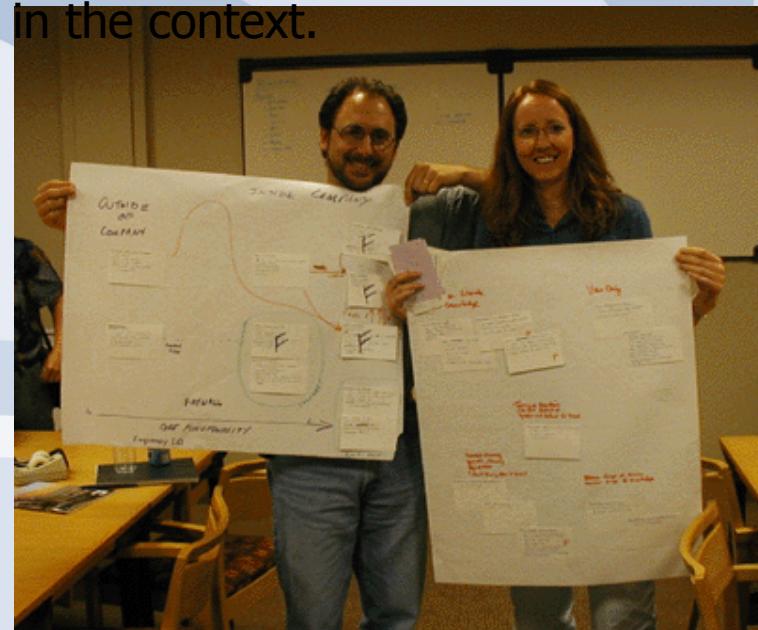
Identify relationships between tasks.



The Process: Seeing the Software Emerge

- **3 x 5 card arrangements of tasks will clump up based on affinity of the tasks.**
- **For each clump of tasks identify an interaction context.**

An interaction context is a “place” in your software, like a room in a house. This is the physical screen or group of screens users will navigate to to perform the tasks in the context.
- **Using roles, tasks & interaction contexts we can proceed to estimate and schedule work.**
- **Post the models as “information radiators.”**



Proud Modelers



Feed Freshly “Designed” Requirements Forward Into Your Favorite Agile Process

- Tasks generated can serve as:
 - XP Stories
 - Scrum Backlog
 - FDD Features
- Focal roles and tasks make priorities easy to identify.
- As needed, add detail to task cases and design user interface just-in-time.
 - U-CD task cases represent a simple dialogue between user and system
 - Using canonical components, it's straight forward to "map" task cases to abstract user interface easily used to validate functionality
 - Render wireframe UI from abstract UI.
- Test abstract UI by role playing the user role performing the task case.



The Tomax development environment uses planning and development practices from XP then incorporates U-CD for initial project scoping and daily functional design refinement.

Conclusions:

- **Agile U-CD is a simple collaborative process that:**
 - Helps identify the users whom the software will serve and tasks they will perform.
 - Helps us identify importance & priority of those people and tasks.
 - Helps us select “behavior and function.”
 - Builds domain understanding and end-user empathy within the team.
 - Feeds other Agile development methods well.
- **U-CD as an instance of Interaction Design is simple, teachable and repeatable.**

Makes adding Interaction Design sensibilities to the team, and project easy.

Agile User Experience Development in a Large Software Organization: Good Expertise but Limited Impact

Kati Kuusinen, Tommi Mikkonen, and Santtu Pakarinen

Tampere University of Technology, Tampere Finland
{kati.kuusinen, tommi.mikkonen, santtu.pakarinen}@tut.fi

Abstract. While Agile methods were originally introduced for small, tightly coupled teams, leaner ways of working are becoming a practical method to run entire enterprises. As the emphasis of user experience work has inherently been on the early phases before starting the development, it also needs to be adapted to the Agile way of working. To improve the current practices in Agile user experience work, we determined the present state of a multi-continental software development organization that already had a functioning user experience team. In this paper, we describe the most prevalent issues regarding the interaction of user experience design and software development activities, and suggest improvements to fix those. Most of the observed problems were related to communication issues and to the service mode of the user experience team. The user experience team was operating between management and development organizations trying to adapt to the dissimilar practices of both the disciplines.

Keywords: User experience (UX); Agile development; human-centered design (HCD), human-computer interaction (HCI)

1. Introduction

User experience (UX) plays a significant role in the success or failure of contemporary software-centric products and services, as well as the companies producing them [15]. The fundamental goal of UX work is to create software that is highly usable and fulfills user needs. UX, standardized as a “person's perceptions and responses resulting from the use and/or anticipated use of a product, system or service,” is inherently based on the iterative model of human-centered design (HCD) [16], where context of use and user requirements are specified, and software is developed to meet those requirements. Yet, most prevailing Agile software development methodologies do not give clear guidance on how to incorporate user experience design activities in the software engineering practices. For instance, the Agile Manifesto [1] itself ignores UX-related activities.

In general, Agile development [1] refers to using iterative and incremental approach that emphasizes collaboration, feedback and working software over

processes, documentation and strict plans. By Agile UX we refer to work that systematically results in desired user experience of the outcome and is conducted according to Agile principles.

This paper describes the state-of-the-practice of Agile UX work in a large software organization that provides specialized software systems, mainly for internet service provider (ISP) markets. The company, one of the front-runners in the Agile transformation on the global scale, has several sites on various continents, and these all work together in an Agile manner. While their software development is aligned with Agile practices, the company has had problems in integrating UX work with Agile development. Our aim is to improve the current situation in Agile UX work within the company, and more generally to reveal good practices in Agile UX. Moreover, the goal is to uncover impediments and issues (both organizational and methodological) that support Agile UX inside the company. The study consisted of a web survey with 50 questions and 76 respondents which was followed by 13 face-to-face semi-structured interviews. We also studied internal documentation and compared it with the results of the interviews and the survey.

The rest of the paper is structured as follows. Next, in Section 2, we provide some related work and background on user experience and Agile development. In Section 3, we introduce the ways of working in the organization in which the study has been carried out. In Section 4, we discuss how the actual study was implemented, and in Section 5, we list the main lessons we have learned from the study. In Section 6, we give an extended discussion of the study and possible future actions. Towards the end of the paper, in Section 7, we draw some final conclusions.

2. Background and Related Work

Previous research has indicated there is a clear need for further studies of UX work in the field of Agile development [12, 25]. Silva da Silva et al. [25] conducted extensive literature research regarding HCD (including UX) and Agile methods. They presented 58 research articles found relevant in their study, and pointed out that most studies have been descriptions and conclusions of experiments. Therefore, they conclude that further studies are needed to understand the best practices in Agile UX work.

2.1 Practicing Agile UX Work

There is no clear consensus on best practices in Agile UX work – in other words, UX work that is conducted in accordance with Agile principles and methods, thus integrating (or merging) UX work and HCD practices with Agile development practices. There is clear evidence that some or little design upfront (SDUF or LDUF, respectively) is needed also with an Agile approach; this is recommended in 31 papers analyzed by Silva da Silva et al. However, the reason those papers recommended LDUF is that big design upfront is against Agile principles, and these articles present no recommendations on which design practices should be used during design upfront. Besides the need for SDUF, another evident finding is that integrating UX design and

other development activities improves communication and collaboration between the functions; this is reported in 26 articles presented by Silva da Silva et al. Common practices in Agile HCD work include low fidelity prototypes, user tests, and user stories. However, ways of utilizing these practices vary. User testing on paper prototypes and on working software were equally recommended [25].

Common problems in Agile UX work include power struggles, differences in schedules between HCD and implementation, communication issues, unwillingness to understand project needs, and failure to achieve the right amount of user involvement [6]. Budwig et al. [4] report problems in understanding the big picture. UX work conducted within a development sprint caused confusion and problems in communication and off-shore coordination [4]. Petrovic et al. [22] state that in many cases, UX specialists are those who end up in quality assurance work. The above issues still remain unsolved, since they are considered as too expensive or difficult to improve. In addition, design vision and information architecture ought to be understood before starting implementation, and their realization assured in development. Such practices prevent the situation in which problems are discovered only after implementation [22].

Ferreira et al. [11] observed that upfront design improves user satisfaction and product consistency. Upfront design assisted in finding affordable design solutions. It also supported project-level estimation and prioritization, and thus led to savings in time and costs. Federoff and Courage [10] report that Agile UX work has been improved by parallel development and design, working one sprint ahead, interactive prototypes for communicating design, and design studios. Budwig et al. [4] report that a UX team benefitted from close collaboration with a broad cross-functional team since issues were found earlier and addressed faster. They also recommend that the UX team should work one to two sprints ahead of development teams in their own sprints, and should be co-located with development teams. The UX team should also work with business people to define requirements before starting UX work. Moreover, the UX team should already be involved in roadmap work. Working closely together helps in sharing information, and starting UX work early enables recognizing and considering user expectations in time, that is, when UX still is affordable and leads to optimal savings in development time and costs.

Chow and Cao [7] conducted multiple regression analysis on 48 common hypotheses of success factors in Agile software development and determined that only 10 were supported. From those, the authors found that the truly critical success factors were these: correct delivery strategy, proper practice of Agile techniques, and a competent team. The second set of important factors included good Agile project management process, a cooperative orally communicating team, and strong customer involvement.

Currently, one of the most recommended process models for applying UX work in an Agile context seems to be the “one sprint ahead” approach (Figure 1) originally proposed by Miller [21] and Sy [27], later applied or modified by Fox et al. [13], Silva da Silva et al. [25], and many others.

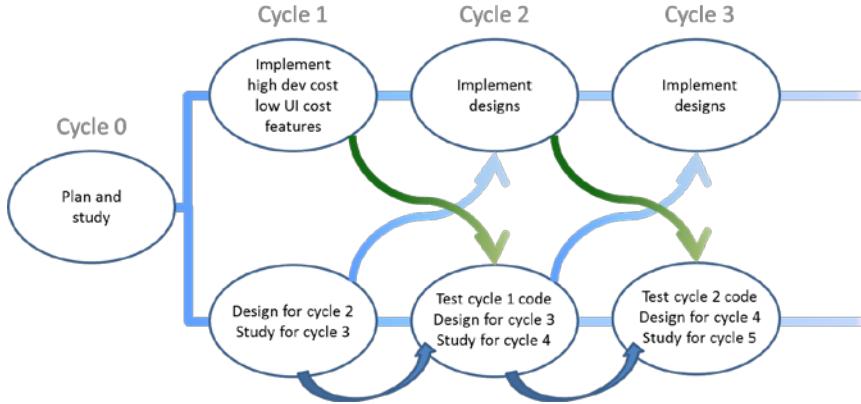


Figure 1. One sprint ahead approach. Redrawn from [27]

The process model of Miller [21] and Sy [27] has been in use at Autodesk, but no recent experience reports were available. One sprint ahead is also recommended by Budwig [4] and Federoff and Courage [10]. Indeed, as by definition, UX design is iterative [16], working at least one sprint ahead gives an opportunity for iterations. Sy [27] observed the approach helped to complete the design just in time, which resulted in less design waste. The approach helped to gain a shared vision; through daily contact developers could be aware of design progression and give their input early enough [21]. Miller found the close cooperation and daily interaction between developers and UX designers to be an essential success factor. She reported savings in design and development time and improvements in user satisfaction [21].

3. Way of Working in the Organization

In the following, we introduce the ways of working within the software development organization we have studied. First, we discuss research and development, where Agile practices play a major role. Then, we address certain other functions of the company, where other practices exist.

3.1 Research and Development

As the company being investigated is one of those that have been spearheading the Agile transformation, it is only expected that Scrum [24, 26] is extensively used in research and development. Scrum (Figure 2) is a simple, iterative framework for project management. When using Scrum, incoming requirements to be implemented are stored in Product Backlog (PB). They are implemented in terms of Sprints – iterations of a fixed length of two to four weeks. For each Sprint, a collection of Product Backlog items are selected and refined into Sprint Tasks. Then, these are implemented in tasks of the Sprint. After each Sprint, a complete system is available, that can be delivered to clients, at least in principle.

Only three roles are defined in Scrum: Product Owner (PO) is responsible for managing the Product Backlog; Team, consisting of developers, is responsible for executing the Sprint; and Scrum Master will eliminate any emerging impediment and is responsible for enforcing the Scrum process. At the heart of the Scrum process are the self-organized Scrum Team and committed involvement of PO. In addition, team stability is of crucial importance, as it is the Team that takes care of the implementation tasks as a single, high-performing and effective entity.

The company being studied has been following Scrum for six years. Due to the size of the company, they also utilize Scrum-of-Scrums, which coordinates the effort of all Scrum teams towards a single goal that is more comprehensive than that of any of the Scrum teams themselves.

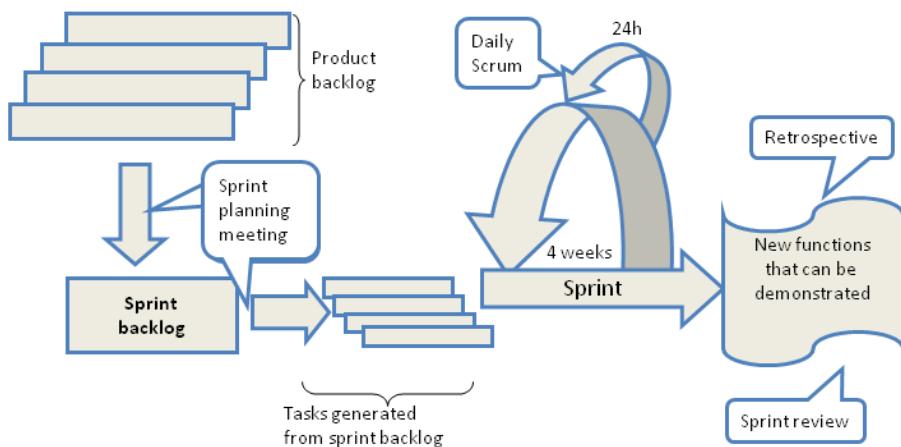


Figure 2. Scrum process

3.2 User Experience

The company has an established UX team with about 15 UX professionals. The team has existed for four years. The majority of the team is physically co-located in Finland, but some of the team is located in Asia. Organizationally, the majority of the UX team is allocated to product management, which is responsible for developing a concept of software before starting a project. Hence, UX specialists are usually involved in development early on. The UX team members have specialized roles. One is specialized in research activities, another is responsible for official communication inside the organization, and some are interaction or graphic designers. The UX team rarely works together as a team. Instead, members have certain projects or tasks they are working with as individuals or sometimes as pairs.

Fixed quarter and yearly plans structure product management's work whereas R&D is following Scrum in their work. In general, product management is responsible of defining the product and making early concepts. R&D implements the products. As the UX team members operate between R&D and product management, their ways of working have characteristics of both product management and R&D – in general, the

UX team's iterative practices do not follow Scrum. While the UX team members usually have some design upfront time, in some cases they start design work as implementation sprints begin. Generally, they adapt the practices of the function or team with which they are cooperating with.

Generally, UX issues are not in PB as such, and there are no acceptance criteria for UX work. However, UX team members working in projects follow PB. Usually developers get the design outside PB as wireframe or high-fidelity images. The UX team has earlier tried Agile working methods inside the team, too, but they were unable to stay one sprint ahead of development. Therefore, the trial was cancelled.

3.3 Other Parts of the Organization

Inside the organization, the use of Scrum is limited to R&D. In contrast, when delivering complete system-level solutions, for example, commonly composed out of the parts that are developed using Scrum, the process resembles a more conservative iterative approach, where releases are planned in advance according to the needs of the most important customers. The approach is iterative but it has fixed scope and time frame. In addition, there are certain management activities that do not follow Agile principles. Such topics fall beyond the scope of this paper.

4. Research Method

The explorative case study [18, 23] described in this paper was conducted by two researchers from a university research organization over a ten-month period from February to October 2011. The main research question was 1. How can the present state of Agile UX work be improved in the case organization. Other research questions were 2. What are the significant challenges in Agile UX work in the organization and how those can be resolved, and 3. What are the current good practices in Agile UX work. Since the case study was explorative, we did not have a hypothesis. We are also studying other companies with similar approach, e.g. [17].

4.1 Main Themes of the Case Study

The present state analysis was conducted in two parts: In the first part, a survey concentrated on collaboration and interaction, processes and tools, and respondents' knowledge about UX and usability; in the second part, a series of interviews concentrated on processes, collaboration, and acquiring and using user feedback. Processes and tools included both official and unofficial practicalities and working methods in the organization, such as the official development process and how employees follow it in practice. The collaboration and interaction section concentrated on how employees communicate with each other, which collaboration methods they would prefer, and what is hindering their cooperation. Knowledge was measured by

how respondents defined UX and usability and how they would rate their own expertise.

4.2 Survey

The web survey was designed to reveal the present state of UX work in Agile organizations in general (in this paper we focus on one particular organization). It consisted of 50 questions (31 open and 19 closed-ended) on collaboration and interaction, processes and tools, and concepts and knowledge. The questions were generated iteratively by two researchers, based on a variety of studies [3, 5, 7, 9, 14, 20, 28, 29, 31]. References were studied to determine significant areas and common problems to generate appropriate questions. Questions and concerns were collected from the mentioned studies and iterated into a survey by the two researchers. The survey was piloted in two companies and iterated based on the feedback before starting the data gathering for the study presented in this paper.

The survey was open for answers for three weeks in February 2011. Invitations to participate in the survey were sent to members of the R&D organization and to its interfaces, such as marketing and management, including the UX team. Interviews were conducted two months later, after the survey had been analyzed. Interview questions were generated to go deeper in the focus areas, and from topics that remained unclear upon analyzing the survey responses.

Open-ended questions of the survey were analyzed by two researchers with a qualitative, theory-bound analysis method with an emergent coding approach [18] as follows. First the researchers separately read the material and classified the data per question. After that the researchers discussed the categories and formed classes. Then the data were coded into classes individually by two subjective coders to quantify them. An inter-rater reliability analysis using the Cohen's kappa statistic was performed to determine the consistency among coders. Quantified data were analyzed with SPSS to find statistical significances. Theory was bounded to data in the analysis phase when linking findings with Agile and UX methodologies.

4.3 Interviews

Semi-structured interviews were conducted as individual or pair interview by one or two interviewers. About 70 to 80 minute interviews were recorded and analyzed from written transcripts. Interview questions covered the job content and tasks of each interviewee: which tasks are their responsibility, who they are working with, and how they communicate with UX specialists and users of their work output. Additionally, questions about utilized Scrum roles and practices were asked.

Interviewees were selected to cover the most focal roles in software development. The company contact person helped in arranging the interviews. Questions were grouped based on the respondents' role. Some of the questions were aimed at all the respondents, but each role also had role-specific questions.

The interviews were analyzed by using the affinity wall method as instructed by Beyer and Holtzblatt [2]. The data-driven method utilizes theory-independent coding on a physical wall. Transcribed interview data were worked into 1126 notes each containing one piece of information. Three researchers (two subjective and one objective coder) categorized notes either including one into an existing category or creating a new one if no suitable category was available. If the note was considered irrelevant to the study, it was excluded from the analysis. Later we labeled and revisited the existing categories, and grouped those under new upper-level categories.

5. Findings

Over the course of the study, it quickly became apparent that although the original goal was to study the relation between Agile development and UX design, the findings would also reveal numerous other organizational aspects. In the following, we introduce the sample and our main findings organized into four categories (understanding UX, UX team, and cooperation). While listed separately, the issues are often intertwined with each other, and improving the situation requires considering them all.

5.1 Survey Participants and Interviewees

Table 1 describes the job roles of the 76 survey respondents. They had been working for the company 2 to 16 years, mean 7 years. Total work experience varied between 4 and 24 years, mean 14 years. Thirty-three respondents (developers, architects, scrum masters, and POs) were from R&D, whereas the rest (38) were from other functions such as management and UX. We calculated Cohen's kappa for coding open-ended answers. The inter-rater agreement was found to be kappa = 0.885 with p < 0.001 (almost perfect).

Table 1. Survey respondents' job categories (open-ended question)

Job	N	Job	N
Developer	20	UX specialist	6
Management	19	Scrum master	4
Product manager	7	Product owner	3
Architect	6	No answer	5
Quality assurance	6		

Table 2 presents office locations of the respondents. All the interviewees and 56.9% of the survey respondents work at the head office in Helsinki, Finland, and thus the core of the findings is strongly linked with that site. Different sites have their own ways of working; there is no general process that all the sites would follow.

Table 2. Office locations of survey respondents (closed-ended question)

Office location	N	Office location	N
Helsinki	37	Kuala Lumpur	7
Bordeaux	9	St Petersburg	3
Oulu	7	Paris	2

Altogether 13 persons with different roles in development were interviewed. Table 3 presents interviewee roles and codes that are used in this paper when presenting results to reveal which of the interviewees are behind each finding. The interviewees had total work experience from 9 to 31 years, mean 15.1 years. They had been working for the company from 3 months to 17 years, mean 5.2 years. And they had working experience of Scrum from 2 to 5.5 years, mean 3.4 years.

Table 3. Interviewees' work roles and codes used in this paper

Code	Role	Code	Role
H1, H2	Product owner	H8, H9	Scrum master
H3, H4	UX designer	H10	Quality engineer
H5, H6	Architect	H12	UX manager
H7	Developer	H11, H13	Product manager

5.2 Overview of the Results

The survey revealed that problems in Agile UX work were mostly process and communication related. The expertise both in Scrum and UX was generally good. Of the survey respondents, 85.7% evaluated their expertise in UX on a six-level scale from inadequate (1) to adequate (6) to successfully complete their assignments four or higher, N=70. On a six-level scale from novice (1) to expert (6), 63.0% evaluated their expertise in UX four or higher. Scrum masters (75.0%), quality engineers (50.0%), product managers (42.8%), and managers (38.9%) considered their expertise in UX low most often. The biggest problems in Agile UX work were considered to be lack of cooperation between UX specialists and developers, lack of time for designing UX / getting UX designs for implementation too late, and balancing the amount of early concept creation work for understanding the big picture before starting a development project.

Interviews concentrated on the process, communication, and acquiring and utilizing user information and feedback. Both the survey and interviews indicated problems in communication due to lack of face-to-face time. In interviews we discussed through the development process and found improvable issues from every phase, such as lack of common practice for early UX work and verifying the implementation of UX design. It was also hoped that UX work should be more agile.

5.3 Understanding UX

We asked in the beginning of the questionnaire how the respondents understand UX in general. UX was understood to be a feeling (28.6%) a user gets when interacting with a product; and the feeling is built on usability (22.9%) (i.e., ease of use and efficiency). Also 22.9% of the 70 respondents emphasized the holistic nature of UX and described it as a whole-life cycle from buying to end of life and said that it includes many aspects. Especially product managers (42.9% of them) and UX specialists (33.3%) referred to the holistic nature, whereas managers (36.8%) and developers (35.0%) described UX as a feeling the user gets while using the product.

Later in the questionnaire we asked what good UX means in the applications or services the participants were developing and got 49 responses. Traditional usability factors were ranked high (Figure 3). Ease of use (34.7%) and efficiency (22.4%) were seen as the most important: 14.3% of respondents thought the service should satisfy user needs, and 14.3% described mainly non-negative user feelings – “*user does not get frustrated.*” Only three respondents (6.2%) described positive feelings, such as “*good UX means that user is happy or likes the application.*” However, more ambitious UX goals are set in the beginning of a project. There is a strong feeling that these goals do not spread in the organization effectively and that people both in the R&D and outside of it are unaware of them. On the other hand, for most of the applications and services the company produces, the product strategy emphasizes invisibility and ease of use.

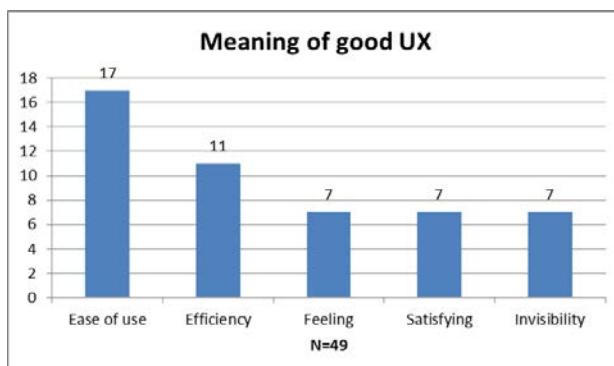


Figure 3. Good UX in our applications (open-ended question)

The company had not defined how they understand UX. Good UX may have various forms depending on context and service. When asked “how does ‘company name’ define user experience,” 37.5% said they do not know; 33.3% described some definition such as ease of use, “*the UX team has defined it,*” or “*no customer complaint means it is ok*;” and 6.3% answered there was no common definition.

5.4 Tasks of the UX Team

The UX team was partly operating as a service organization where product managers or developers could order some design work. It seems that service mode was causing

many of the observed problems related to UX work, such as inability to constantly deliver design in time for development or doing UX work too late. Lund [19] claims service mode is the biggest threat to viable UX work. In the survey, 33.3% of respondents answered that one of UX team's most important tasks is consulting or helping development teams (Figure 4). A UX team member clarified this: “*When a project has remembered they need some specs or screens, then they have ordered those and we made them.*” On the contrary, the UX team told in interviews that their biggest responsibilities are making designs and understanding the big picture. Six interviewees wanted to see UX specialists giving guidelines and making high-level designs whereas the developers would solve design details and trivial cases by themselves with help from UX team when needed (H1, H3, H4, H9, H11, H12).

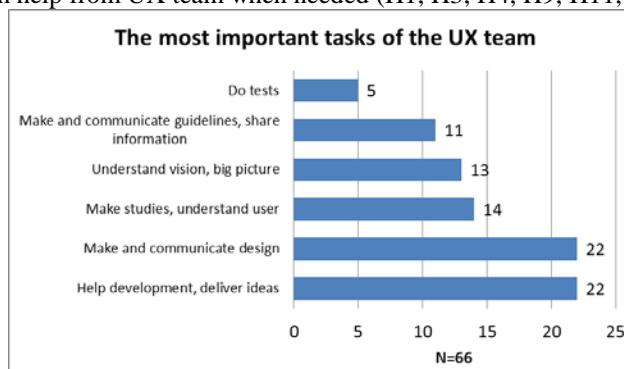


Figure 4. In the survey, the most important tasks of the UX team were considered to be to consult and design (open-ended question)

The UX team seemed to have limited power of decision: “*One of the most challenging issues in our work is that we have lots of responsibility, we should manage the overall picture of all the requirements and how it can be captured in the UI level. And on the other hand, we have no power of decision at all; we need to be diplomatic with everyone to find the happy medium*” (UX specialist). When we presented the results to company employees (mainly product managers, product owners, UX specialists, and architects), they considered understanding vision, the overall picture, and user should be more important.

5.5 Cooperation with UX Specialists

In general, the UX team operates as a link between R&D and product management. When asked ‘*what is expected from you in your job?*’ an interviewed UX specialist told “*it is mostly communication, acting as a link between development and product managers. And to visualize both business and technical requirements and try to transfer those into pleasant, easy-to-use user interfaces.*” Physically the majority of UX specialists are co-located with product management and apart from R&D. Survey respondents considered in an open-ended question that at its best UX cooperation is started early (18.6%), and communication happens face-to-face (15.3%). However, 11.9% reported cooperation would be at its best as long as it actually happened, or if

they just had a UX specialist available. It seems there are problems with availability of UX resources.

In contrast, unsuccessful cooperation (Figure 5) is conducted too late (26.7%), there is too little or no communication (18.3%), or the design the UX team delivers is unsuitable or not implementable (18.3%). Unclear responsibilities or disagreement on who can make decisions hinders cooperation (13.3%). Too little communication between organizational units may lead to such non-Agile practices as working separately and communicating ready-made work between units (e.g. between UX and R&D). In general, more agile approach for UX work was hoped for (H1, H2, H3, H4, H7, H8, H9, H10, H13).

Participants from R&D generally wished for more face-to-face time with the UX team (Figure 6), as observed both in the survey and interviews. Synchronizing R&D and UX work was seen as a major problem as UX and R&D frequently work on different schedules. Interviewees from R&D mentioned getting contributions from the UX team (e.g., for design) takes too much time. In addition, UX and R&D should cooperate more frequently in an iterative manner. The interviewed architects hoped that architecture and UX design could be done in iterative cooperation.

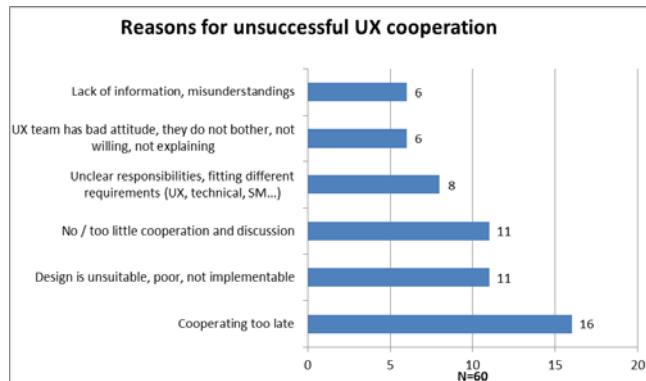


Figure 5. Unsuccessful UX cooperation happens too late (open-ended question)

Often developers are introduced to the design only when implementation starts. Typically, product management outlines the first draft of a concept, and often the UX team is involved, whereas a development team usually joins the work later. “*When the development team gets involved, they just say ‘no!’ We should have discussed earlier.*” Similar problems occur when product management makes first drafts with developers and without UX team, “*Sometimes all parties are involved (from the beginning), but sometimes product managers work directly with developers and we (UX team) are like ‘no,no,no! UX suffers’.*” Interviewees described there is no guidance on cooperation available; individual product managers decide if they want to include UX issues, and UX work can be easily cut out (H1, H3, H5, H6, H10, H12).

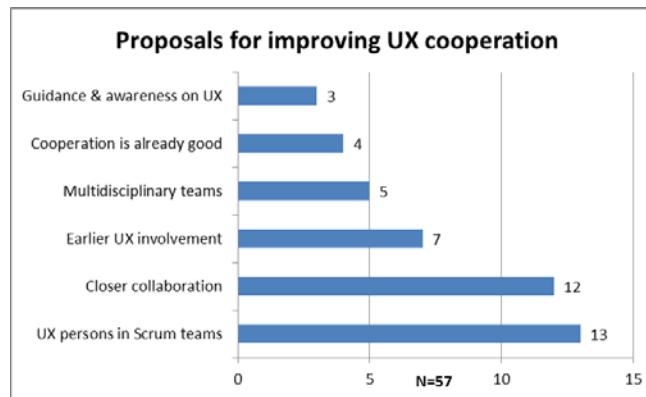


Figure 6. Respondents hoped for more constant and frequent cooperation with UX specialists (open-ended question)

In many cases, architecture designs are made without feedback from the UX team and architects have to “guess” how it should be: “*It's better to have some UX input before final design.*” Interviewees said that the UX team gives feedback on ready-made architecture design and says that this should have been done differently. The same occurs with UX design: “*It kind of feels that we just get some ready-made (UX) design at some point and this is how it looks, and there's no chance to affect that.*” Both interviews and survey indicated that when there are incompatibilities between UX and architectural design, UX and architects discuss changes to architecture with the product owner and product manager. Changes are made if there is time, but usually architecture stays as it is.

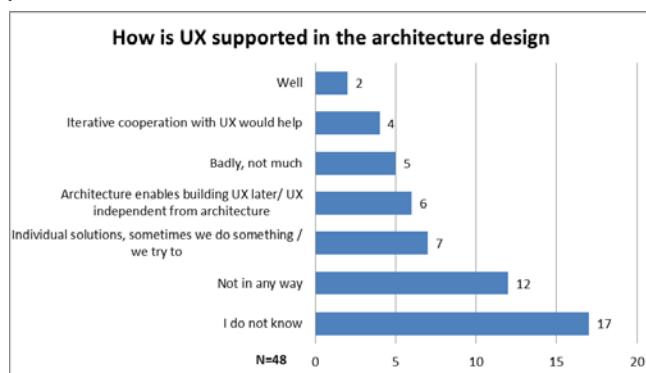


Figure 7. UX is not present in architecture design (open-ended question)

Architecture may be designed without the UX team’s contribution even in those cases where contribution would be needed. Over third of survey respondents (35.4%) did not know how UX is supported in architecture design (Figure 7). One fourth (25.0%) of respondents say it is supported minimally or not at all; 14.5% describe their own informal solution; 12.5% believe good architecture enables building good UX later on or that UX is not affected by architecture; and 8.3% hope to have iterative cooperation between UX specialists and architects in the future.

There are no commonly agreed-upon practices in, for example, communicating design or documenting user requirements. Depending on the development team, the UX team may deliver design e.g. as wireframes or highly detailed Photoshop images. Again, at worst, ready-made UX designs may be communicated via email to developers. User requirements may be communicated as user stories or use cases (34,5%), or by various other means. Of the survey respondents, 54,5% told user requirements are documented informally, poorly, or not at all. Interviewed architects and developers hoped user requirements would be incorporated and explained in user stories. Interviewed UX specialists told they would prefer doing wireframe that developers would complete by following the style guide. Doing detailed Photoshop images was seen as waste of time since the details can be seen from the style guide. However, currently it is not ensured that UX will be implemented according to the style guide, or as designed. The interviewed UX specialists hoped that developers or quality assurance would ensure the UX implementation.

6. Discussion

In general, the company had a functioning and competent UX team with established practices. However, the company had divergent ways of working, and UX work was not included in the development process and practicalities sufficiently enough. There were many unofficial practices concerning UX work. Individuals could have made their own decisions whether to follow those practices in their work. Especially decisions made by individual product managers and product owners had a significant effect on the impact of UX work in the company.

Currently, UX specialists had an iterative way of working where they did the design work based on an existing product backlog. They were rarely involved before the PB was created. In some cases UX specialists had some design upfront time. But regularly they also had to start the design work only after implementation work had been started, which they considered stressful. UX design was communicated to developers with various methods depending on developers and the skills of the UX specialist. The UX team also had the role of quality assurance in UX issues as no one else ensured that implementation follows the style guide or the UX design. One explaining issue might be that half of the quality engineers who participated in the survey considered their expertise in UX issues low.

6.1 Identified Problems and Potential Solutions

Table 4 presents problems the organization had with UX issues and suggests solutions to fix those. The solutions are based on related work and the development and business models utilized in the organization. Thus, suggestions from literature are compounded with the present state of the organization. The organization used the suggested solutions to evolve their current practices and to create new better ones. Mainly, the issues presented in the table are due to unclear practices and responsibilities, and the service-organization approach that was prevalent in the

company. The organization already has a functioning and efficient UX team; the majority of UX work already leads to a good outcome. These practices are part of continuous improvement activities and a means to further systematize UX work.

Table 4. Suggestions for observed UX challenges.

Problem	Solution
UX work is started too late	UX specialist involving roadmap concept creation and UX related decisions (early involvement suggested by e.g. [4, 8, 10, 25])
UX design does not match architecture design	Both UX and architecture design made iteratively with both disciplines involved [4, 22]
Developers misunderstand design, UX team does not notice technical limitations	A developer participating in UX design [4, 22]. The participating developer as the first contact person for other developers
UX specialists have to hurry with design	Design made in UX sprints with the help of an architect or developer [4, 22], one sprint ahead of development [4, 10, 21, 27]. Some design upfront [11, 25]
UX design is not ensured during development	Acceptance criteria for UX issues; team owns UX implementation, PO or quality engineer approves [22]
UX work is bypassed because of tight timeline	Criteria for minimum/desired UX at project start [30]
UX elements dropped arbitrarily during implementation	Plan for minimum UX design realization; essential and optional parts or solutions [29]. Power of decision for UX people [6, 11]
UX team doing lots of reactive work	UX team involved earlier [4, 8, 10, 25]. UX specialist involved in decisions when and where UX work is needed [6, 11]. UX team not a service organization [19]

6.2 Implementing Results in the Organization

A workshop reviewing the study results was arranged in the company after the study period during fall 2011. The 14 participants were from different areas of product development, both from R&D and outside of it (3 product owners, 2 architects, a scrum master, a quality engineer, 3 product managers, 3 UX specialists, and a director responsible of R&D methods). The workshop aimed at generating ideas for concrete actions that could be taken to improve the current situation. The participants worked in three groups of four to five persons, and they were instructed to create seven realizable ideas and select three of them to present to others. All the groups wanted to ensure that UX design is realized during development, and therefore suggestions were limited to seven. Finally, the presented ideas were ranked by voting. The ranked actions were as follows:

1. Product Backlog items will be groomed for UX before development starts.
2. Product, architecture and UX roadmaps will be synchronized regularly.
3. UX target will be set in the beginning.
4. Release UX quality will be fast-checked during development.
5. Design version control will be improved.

6. UX resources will be increased.
7. UX stakeholder will be in action during project.

Later, a UX review method was implemented in the official development process. With help of others, the UX team created the method to tackle the most severe issues found in the study presented in this paper: The company started a monthly practice where a user study is arranged to examine a selected feature or product. The participant roles are usually two UX specialists, UX manager, a product manager, a product owner, an architect or developer, and a user. The user has pre-defined tasks to perform, and the others are observing and making notes of certain issues. After the user test part of the meeting, the test is revisited task by task and everyone marks on a whiteboard the biggest flaw they observed. The observations are discussed and ordered. The product owner and architect or developer orders the list by business value and developer effort. The product owner selects how many of the list items they can commit to within the next release cycle. The first experiences of the method show that the method helps to reveal UX flaws and increases the participants' understanding of UX. However, the fixed release plans strongly limit the agility of R&D. In one UX review meeting that a researcher was observing during spring 2012, the product owner said they are able to welcome none of the items to the product backlog since they are already too busy with the release.

7. Summary and Conclusions

In the study described in this paper, we conducted a present state analysis concentrating on UX within Agile software development in a large software organization.

76 employees participated in a large, mainly qualitative survey and 13 persons with different work roles were interviewed. The data were analyzed with a qualitative content analysis method, and with SPSS. Results were divided into categories based on the survey structure, that is, into process, cooperation, and understanding of UX. Another category, the UX team, was generated during analysis mainly to describe the team's current working mode and position in the organization.

The organization is advanced both in Scrum and UX work; they have a competent UX team and knowledge about both Agile and UX development is at high-level in the organization. However, the organization has a separate product management (including the UX team) which operates by yearly and quarter plans, and R&D which follows Scrum. Differences in schedules and ways of working make cooperation more challenging. The most pervasive challenges were related to communication problems, too late UX work, and issues in power relations and in areas of responsibility. Communication issues, caused mostly by too little face-to-face time, led into misunderstandings and time wasting. UX specialists' insufficient power of decision enabled e.g. product owner, product manager or development team to cut out UX issues generally at any point. Responsibility issues seem to be common in Agile UX as the role of UX specialists is unclear in many cases.

The results are not conclusive in the sense that they are to a great extent addressing organizational rather than methodological issues. Consequently further research in this field is needed in order to refine Agile methodologies to better support UX related issues.

8. Acknowledgment

We want to thank all the participants of the study: the interviewees and survey respondents, and persons who read this paper and gave their valuable comments. In particular, special thanks go to our contact person in the company for the commitment to our study. Finally, the financial support of the Finnish Funding Agency for Technology and Innovation (TEKES) is gratefully acknowledged.

9. References

1. Agile Alliance. Manifesto for Agile Software. Available at <http://agilemanifesto.org>. (2001)
2. Beyer, H., and Holtzblatt, K. *Contextual Design: Defining Customer-Centred Systems*. Morgan Kaufmann. (1998).
3. Boivie, I., Gulliksen, J., and Göransson, B. The lonesome cowboy: A study of the usability designer role in systems development. *IWC 18*, 4, 601-634. (2006).
4. Budwig, M., Jeong, S., and Kelkar, K. When Usability met Agile: A case study. In Proc of the 27th international conference extended abstracts on Human factors in computing systems (CHI EA '09). ACM, 3075-3084. (2009).
5. Bygstad, B., Ghinea, G., Brevik, E. Software development methods and usability: Perspectives from a survey in the software industry in Norway. *Interacting with Computers* 20, 3. 375-385. (2008).
6. Chamberlain, S., Sharp, H., and Maiden, N. Towards a framework for integrating Agile development and user-centred design. LNCS 4044, pp. 143–153. (2006).
7. Chow, T., and Cao, D.B. A survey study of critical success factors in Agile software projects, *Journal of Systems and Software*, 81 I 6, 961-971. (2008).
8. Coplien, J.O., and Bjørnvig, G. *Lean Architecture: For Agile Software Development*. John Wiley and Sons. 376 pages. (2011).
9. Earthy, J. Usability maturity model: Processes. [http://www.idemployee.id.tue.nl/g.w.m.rauterberg/lecturenotes/Usability-Maturity-Model\[2\].pdf](http://www.idemployee.id.tue.nl/g.w.m.rauterberg/lecturenotes/Usability-Maturity-Model[2].pdf) (1999)
10. Federoff, M. and Courage, C. Successful user experience in an Agile enterprise environment. LNCS 5617, 233-242 (2009).
11. Ferreira, J., Noble, J. and Biddle, R. Up-Front interaction design in Agile development. *Lecture Notes in Computer Science* 4536, 9-16. (2007).
12. Ferreira J., Sharp H. and Robinson H. User experience design and Agile development: Managing cooperation through articulation work. *Software Practice and Experience*; 41,9, 963-974. (2011)
13. Fox, D., Sillito, J. and Maurer, F. Agile methods and user-centered design: How these two methodologies are being successfully integrated in industry. *Proc. AGILE 2008 conference*. IEEE Press, 63-72. (2008).
14. Hussain, Z., Slany, W. and Holzinger, A. Current state of Agile user-centered design: A survey. LNCS 5889, 416-427. (2009)

- 15.Innes, J. Why Enterprises Can't Innovate: Helping Companies Learn Design Thinking. In *HCII 2011, LNCS*: 6769, 442–448. (2011)
- 16.ISO 9241-210:2010. Ergonomics of human-system interaction. Part 210: Human-centered design for interactive systems (2010)
- 17.Kuusinen, K., Väänänen-Vainio-Mattila, K. How to Make Agile UX Work More Efficient: Management and Sales Perspectives. Accepted to NordICHI 2012
- 18.Lazar, J., Feng, H.F., and Hochheiser, H. *Research Methods in Human-Computer Interaction*. John Wiley and Sons. (2010)
- 19.Lund, A.M. Creating a user-centered development culture. *interactions* 17,3, 34-38. (2010).
- 20.Marcus, A., Ashley, J., Knapheide, C., Lund, A., Rosenberg, D. and Vredenburg, K. A survey of user-experience development at enterprise software companies. *LNCS 5619*, 601-610. (2009).
- 21.Miller; L. Case Study of Customer Input for a Successful Product. Proc. of Agile 2005. Agile Alliance. (2005).
- 22.Petrovic, K. and Siegmann, M. Make space for the customer: The shift towards customer centricity. *LNCS 6769*, 485-490. (2011)
- 23.Runeson, P., Höst, M. Guidelines for conducting and reporting case study research in software engineering. *Empirical software engineering*, 14, 2 2009.
- 24.Scrum Alliance <http://scrumalliance.org/>
- 25.Silva da Silva, T., Martin, A., Maurer, F. and Silveira, M. User-centered design and Agile methods: a systematic review. *Proc. of the International Conference on Agile Methods in Software Development (Agile 2011)*.
- 26.Sutherland, J. and Schwaber, K. The scrum papers: Nut, bolts, and origins of an Agile framework. Available at: <http://jeffsutherland.com/ScrumPapers.pdf> (2011).
- 27.Sy; D. Adapting usability investigations for Agile user-centered design. *Journal of Usability Studies* 2, 3, 112-132. (2007).
- 28.Venturi, G. and Troost, J. Survey on User Centred Design integration in the industry. In ACM International Conference Proceeding Series; Vol. 82. *Proc. of the third Nordic conference on Human-computer interaction*. (2004)
- 29.Venturi, G., Troost, J. and Jokela, T. People, organizations, and processes: An inquiry into the adoption of user-centered design in industry. In *International Journal of Human Computer Interaction* 21, 2. (2006).
- 30.Viikki, K. and Palviainen, J. Integrating Human-Centered Design into Software Development - An Action Research Study in Automation Industry. *Proc. of (SEAA'2011)*, IEEE Computer Society. (2011).
- 31.Zhou, R., Huang, S., Qin, X. and Huang, J. A survey of user-centered design practice in China. In: IEEE International Conference on Systems, Man and Cybernetics, pp. 1885–1889. (2008).

Understanding the UX Designer's Role within Agile Teams

Tiago Silva da Silva¹, Milene Selbach Silveira²,
Claudia de O. Melo³, and Luiz Claudio Parzianello⁴

¹ ICMC/USP - Universidade de São Paulo - Campus de São Carlos
tiago.silva@icmc.usp.br

² FACIN/PUCRS - Pontifícia Universidade Católica do Rio Grande do Sul
milene.silveira@pucrs.br

³ IME/USP - Universidade de São Paulo
claudia@ime.usp.br

⁴ Grupo RBS
luiz.parzianello@gruporbs.com.br

Abstract. User-Centered Design spends a considerable effort on research and analysis before development begins. On the other hand, Agile methods strive to deliver small sets of software features to customers as fast as possible in short iterations. Whereas the two methodologies have tensions regarding requirements gathering and upfront design, they also share similarities. For instance, both approaches are iterative and customer focused. However, there is little guidance on how to integrate these two perspectives and a lack of understanding with respect to the User Experience (UX) Designer's role in an agile environment. Based on four ethnographically-informed studies in two large companies, we aim at providing a better understanding of the integration of Agile development and UX Design by describing the different roles that a UX Designer plays within an Agile environment.

Keywords: Agile, User Experience, Designer, Roles, Stages.

1 Introduction

In an increasingly competitive world, where millions of products compete for attracting users' attention, the User Experience (UX) of a product may determine its success or failure. Agile software development methods have also been proposed for the customer's competitive advantage. Despite having different underlying concepts, Agile methods and UX Design aim at producing high quality software. However, it is known that the integration of UX Design into Agile Methods has not been properly addressed [1]. Agile methods have a distinct culture that at first glance seems to conflict with UX Design [2].

Despite their tensions, they also have similarities [3]. The main similarity is that both approaches are iterative and user or customer focused. Notwithstanding, most of the UX Designers have not been concerned about project

management methods before their first contact with Agile. UX Designers need to be care because, as organizations look for more effective and efficient ways to deliver projects, more and more of them are adopting Agile methods [4].

Frequently, at first glance UX Designers ‘notice’ that there is no design phase in Agile. From the Designers’ standpoint Agile methods strive to deliver small sets of software features to customers as fast as possible in short iterations, implying that design is not a crucial part of the development process. What a UX Designer sees are multiple short deadlines in which working software is delivered and no consideration is given to the many design activities [4].

To the best of our knowledge, there is little guidance on how to successfully incorporate UX Designers into Agile teams. Moreover, there seems to be a lack of understanding regarding the UX Designer’s role in an Agile environment [5]. In this regard, the study herein presented aims at discussing the UX Designer’s Role in Agile teams based on ethnographically-informed studies in two large companies.

This paper is organized as follows: Section 2 presents the background to the problem. Section 3 describes the studies, data collection and data analysis. Section 4 presents the findings and Section 5 discusses some implications and limitations and presents final remarks of this research.

2 Background

User-Centered Design provides specialized skills in User Interface (UI) Design while Agile approaches prefer generalists and discourage extensive upfront design work [2].

Singh [6] proposes an adaptation of Scrum to promote usability. In this adaptation there is the U.P.O (Usability Product Owner) role. The U.P.O. is included in the project effort from the beginning as a peer of a traditional P.O. The two Product Owners work together to first achieve an agreement on the user experience vision for the project. According to the authors [6], the formulation of the vision incorporated the needs of internal customers, developers who have high domain knowledge, and from external customers who would be using the product. Beyer [7] advocates that UX Designers must better understand the Agile principles and presents some practices to the integration of this two fields.

It is not usual to find papers addressing the UX Designer’s role in Agile teams despite there are a bunch of studies addressing the integration of UX and Agile in a higher level. Sy [8] describes adjustments on the timing and granularity of usability investigations, and on how the UX Designer reports his usability findings in an Agile environment. Sy [8] states that the Agile communication modes have allowed them to narrow the gap between uncovering usability issues and acting on those issues by incorporating changes into the product. However, she does not address the different roles that a UX Designers play.

Salah [9] provided a software process improvement (SPI) framework for Agile and User-Centered Design integration with generic guidelines and practices for organizations. Despite this study aimed to achieve this integration, it did not mention the UX Designer roles.

A qualitative study presented by Ferreira *et al.* [10] shows that the nature of iterative development facilitates the performance of usability testing, allowing developers to incorporate the results of these tests in subsequent iterations. They say that this can also significantly improve the communication and relationship between UX Designers and developers, showing hope that these practitioners notice that working more closely may assist them in achieving their common goal.

McInerney and Maurer [2] interviewed UX Designers involved in Agile projects and discuss how UX Designers found their role in Agile environments. According to them, the literature does not identify a distinct UX role, so the onus remains on UX to justify and define its role on the team.

Ferreira *et al.* [10] report some implications for the team arrangements. The authors state that the boundaries between the roles of UX Designers and Agile developers are more fluid in the studies where the UX Designer is considered part of the team than in a study where the UX Designers are not part of the team and did not take part in the sprint planning meetings, standups or retrospectives.

As aforementioned, while these studies reported or proposed principles, adjustments or guidelines, and attempted to merge one method to another, none specifically have addressed the UX Designer's role throughout the agile project cycle. However, this role may change significantly throughout the project or product development and it is highly dependent on the context in which it takes place. Agile development and UX Design emerge from the particular problems that practitioners face in the settings in which they work [10].

Our objective is to provide a better understanding of the integration of Agile development and UX Design. We believe it is crucial to understand the UX designer role in an agile environment in terms of the activities and tasks that should be adopted.

3 Cases Description

Four studies in two large companies were carried out to investigate how a UX Designer works in an organizational agile environment. They are ethnographically-informed [10] and, for instance, instead of spending months or years in the field, we spent the amount of time that fit with the development cycles [11]. By adopting the ethnographic approach, we tried to understand practice in its natural setting with minimal researcher intrusion.

We collect our data by observations, interviews and discussions with practitioners, but do not attempt to change or influence practice during the study. We avoid any form of control, intrusion or experiment and so all the data were naturally occurring, as suggested by [11]. Even our interview data may be viewed as naturally occurring, since it was gathered from practitioners reflecting on their practice in their place of work [11]. Finally, the findings and conclusions were confirmed with the teams members involved.

The next sections describe the organizational setting, the projects, who participated in the studies, how data were collected and how analyses were performed.

3.1 Organizational Setting

In Company 1, the team of developers was one of several Scrum teams in the company working on software development¹. The developers and designers were seated in an open-plan office space located in the same building. However, they were not co-located, *i.e.*, they did not share the same workspace. They were spread in the building, but the UX team members were seated close to each other.

In Company 2 there is no separated UX Team and Developers Team. Each Agile team has its own individuals, *i.e.*, a team does not share developers or UX Designer. These teams were selected because they were the most senior Agile teams in the company. The developers and designers were seated in an open-plan office space located in the same building and in the same floor. Each team is co-located.

3.2 Projects

We followed two projects in Company 1; Project X consists of the development of new features for an existing product of the company. Project Y consists of the development of an existing product of the company for a mobile device.

Company 2 is not structured by projects, but by digital products. It is a digital product-driven business. Two different teams developing two different products were studied. Product X consists of a web portal about agribusiness in the country. Product Y consists of a web portal of services and opportunities in which there are addresses and data from companies and services.

3.3 Participants

In Company 1, our study involved a team of seven individuals and one UX designer. The developers were part of the ‘Development Team’ and the designers part of the ‘UX Team’. The developers had been developing software using Scrum for approximately two years. Although they are called developers, individuals in the team have their own role according to their area and skills. The roles were Project Manager/Scrum Master, Product Owner, Technical Leader, Developer and Tester as presented in Table II.

Information architects, graphic designers and interaction designers compose the UX team. Each project has one UX designer, but a UX designer usually work with more than one development team. The same goes for Project Managers, and they are also known as Scrum Masters in the teams.

The UX member’s role in Project X was to help software engineers to envision new features for this product. In Project Y, the UX member’s role was to prototype and design the User Interface and the User Interaction flow for the product. It is noticeable that the UX Designer plays different roles in different projects, even though in the same company.

¹ The company also develops hardware.

Table 1. Composition of the Teams

Roles played in Projects	Company 1 - Project X	Company 1 - Project Y	Company 2 - Product X	Company 2 - Product Y
<i>Project Leadership</i>	Project Manager / Scrum Master	Project Manager / Scrum Master	Business Owner / Director	Business Owner / Director
<i>Product Leadership</i>	Product Owner	Product Owner	Product Leader / Product Owner	Product Leader / Product Owner
<i>Technical Leadership</i>	Technical Leader	Technical Leader	Scrum Master	Scrum Master
	Developers	Developers	Developers Testers	Developers Testers
<i>Development</i>	—	—	SEO	SEO
—	—	—	UX Designer	UX Designer
<i>Team</i>	—	—	—	Graphical Designer
<i>Supporting Team</i>	UX (shared)	UX (shared)	—	—

In Company 2, our study involved UX designers and their interactions with an Agile team working on the same product. The teams are composed by Product Leader/Product Owner, UX Designer, Developer, Tester and Search Engine Optimization (SEO), with little differences as can be noticed in Table II.

One team – Product X – has two individuals focused on UX, a UX Designer and a Graphical Designer, whereas the other team – Product Y – has just a UX Designer who performs the role of a Graphical Designer as well.

The UX designer's role in Product X was to perform user research, benchmarking and interaction design. The Graphic designer's role was to design the User Interface (UI) based on the wireframes provided by the UX designer. Whereas in Product Y, UX designer used to play both roles, performing user research, benchmarking, interaction design and UI design. Again we may notice the diversity of roles that a UX Designer may play in different teams.

3.4 Data Collection

We used two first-degree techniques [12] for data collection: observation and interview.

In Company 1, regarding observations, due to the characteristics of invoking the least amount of interference in the work environment and the least expensive method to implement and still because the company did not allow video or audio recording of the meetings, we choose to manual record the observations of the meetings.

We shadowed a UX person during his activities for 45 days and observed meetings that he was involved, such as meetings of the UX Team of the company and some meetings of two different projects. We also interviewed three members of the UX group that work in different projects and one project manager, as presented in Table 2. The Project Manager was interviewed aiming to define which Agile Method the company uses and how this integration of UX and Agile works from his point of view. The UX Designers were interviewed aiming to understand UX people work on the different projects of the company. In Company 1, our studies were carried out over three months iteratively.

Table 2. Description of the data sources

Data source	Company 1 Project X	Company 1 Project Y	Company 2 Product X	Company 2 Product Y
<i>Observed Meetings</i>	2 Requirements, 1 Planning	3 Planning, 3 Retrospective, 1 Demo, 5 Daily, 2 User Test Sessions	5 Daily	1 Planning, 1 Retrospective, 5 Daily
<i>Interviews</i>	1 Project Manager / Scrum Master, 1 UX Designer	2 UX Designers	1 Scrum Master / Product Leader, 1 UX Designer	1 Scrum Master / Product Leader, 1 UX Designer

In Company 2, as in the first study, we conducted interviews and observations, manually recording our observations. We observed some meetings of two different teams and we interviewed the UX Designer and the Product Leader of the two selected teams, as can also be observed in Table 2. In this company, our studies were carried out over two iterations – 25 working days. The length of the sprints varies from project to project, but for the two teams observed they have three-week sprints.

3.5 Data Analysis

We analyzed data using the open and focused coding techniques. In the open coding, the researcher reads field notes line-by-line to identify and formulate any and all ideas, themes, or issues they suggest, no matter how varied and disparate. In the focused coding, the researcher subjects field notes to fine-grained, line-by-line analysis on the basis of topics that have been identified as of particular interest [13].

Preliminary memos were extracted from the field notes. Having the memos produced, open coding was performed aiming to generate new insights and

themes. Focused coding was also performed and this coding consisted of linking the memos generated to key aspects identified in a Systematic Review previously performed [5]. In this process, some new aspects emerged from the analysis of the observations and interviews. Later, some integrative memos were written to relate the field notes, the key aspects and the new codes from the open coding. Our findings with regards to the UX Designer roles are presented as follows.

4 Findings

In this section we present our findings regarding the different roles that the UX Designer may play in Agile teams. In each of the subsections below, we identified their responsibilities and skills, and provided some relevant passages from the observations and interviews.

4.1 User Experience Designer

The User Experience Designer role is **responsible** for the understanding of users.

It is desirable that the User Experience Designer has the following **skills**: User Research, Ethnographic Studies, User Experience Design, User Profiling, Ideation, Competitor Analysis, Design Thinking, Customer Journey Mapping.

It is worth mentioning that we based our skills' classification on the skills listed in [4].

The following quotes represent some of the tasks performed by the User Experience Designer in our studies:

“As we have a set of users (database of volunteers), we can call them and carry out some focus groups. We have 4 different personas with them” [C2 - UXB²]

“Some User Research is performed by the Marketing Team. In general, the Marketing Team knows what they say they need, not what they really need. It’s a not a target effort to gather what the user need’ It’s a sell visit.” [C1 - UX1]

“We perform some speculative research, analysis of competitors” [C2 - UXA]

“We have something that we call Discovering that happens before the planning” [C2 - PLA]

The passages above highlight the activities performed by the UX Designer as User Researcher, or as a User Experience Designer itself as we named this role.

We notice activities like benchmarking, conduction of focus groups and definition of personas, for instance. We may also notice that Company 1 has a Marketing Team that provides some data to the UX Team. However, according to their report, the data gathered by the Marketing Team is more about the users' desires than their needs. This observation highlights the need of having a UX Designer carrying out this kind of research. In general, UX Designers are trained to carry out these activities.

² The passages are identified by the Company (C#) and the by the team member of each team interviewed (UX#).

It is noteworthy that this role should work alongside the Business Analyst to create a design vision and design direction from the user experience.

4.2 Interaction Designer

The Interaction Designer role is **responsible** for Designing and Evaluating the users' interaction with products or services, both on prototypes as on the developed system.

It is desirable that the Interaction Designer has the following **skills**: Interaction Design, Rapid Prototyping, User Experience Design, Product Design, Guerrilla Testing Sketching, Usability Testing, Ideation, Collaborative Design, Process Flows, Information Architecture, Service Design, Design Thinking.

The following quotes reflects the tasks performed by the Interaction Designer in our studies:

“We don't need to design everything up front” [C1 - UX3]

“We should work at least one sprint ahead the development team” [C1 - UX3]

“Sometimes we add new user stories based on the results of the User Testing. But it depends on the problem. We also can put as a bug” [C1 - UX2]

“We perform some inspection evaluations, peer review with some UX member” [C1 - UX2]

“We put UX criteria as acceptance criteria at the User Stories, or we reference the behavior of the interface in a sequence of wireframes” [C2 - UXA]

“We perform a lot of informal evaluations. Myself and the Graphical Designer” [C2 - UXA]

By researching from the early stages of the project, the UX Designer may build his own ‘UX Backlog’. Afterwards, as reported by [C2 - UXB], the Interaction Designer may use these data to design or even prototype one iteration ahead of the development team.

We noticed that whenever the UX Designer works close to the Product Owner, they achieve better results on describing business or users' needs. Developers better understand designs and User Stories when they are built by two members with different backgrounds. Further, User Stories become more clear when enriched by wireframes, for instance.

By having designs, sketches or wireframes, UX Designers may start an evaluation process. We observed UX Designers performing informal evaluations, peer reviewing their designs by pairing with other Designers or Product Owners or even Business Analysts. These early evaluations are very important because they avoid future rework and helps to define what will be built. The Interaction Designer may also works alongside the developers to figure out how it can be built.

4.3 UI Developer

The UI Developer role is **responsible** for the Development of the Graphical User Interface (GUI) and the Design of Graphical Elements.

It is desirable that the UI Developer has the following **skills**: Rapid Prototyping, Collaborative Design, Information Architecture, Visual Design, GUI Design, Service Design, Design Thinking.

UI Developers are often the link between the front end and designers as they can speak both languages. As the Interaction Designer, the UI Developer also works alongside the developers and testers to figure out how it can be built.

In our studies, this is the role least played by UX Designers. Most of the UX Designers observed did not develop the UI. In Company 1, for instance, there are few Visual Designers who answer to the teams just when they are required. However, the following passage reveal that developing may not be trivial to all the UX Designers. This happens due to their heterogeneous backgrounds.

“It’s tricky to UX people to code” [C1 - UX2]

In Company 2, one of the teams has a UX Designer and a Visual Designer. The other one has a UX Designer has the skill of visual designing and also performs the Visual Design, as follows: *“Once the product is defined, I prototype it in two or three weeks. Paper prototype to communicate between us and some HTML to present to directors.”* [C2 - UXB]

5 Discussion and Final Remarks

We defined three essential roles that a UX Designer may play in Agile teams. Each of these roles encompasses several skills as described in the previous section.

We do not aim to define these roles as an absolute truth. It is just a simple way of defining UX Designers' roles. In contrast, Ratcliffe and McNeill [4] state that UX Designers may be: User Interface, Interaction, and Usability Designer; Experience Designer; UI Developers and Front-End Developers; Information Architects; Visual Designers; and/or Design Researchers. We assume this fine-grained system of seven role may be too detailed to reflect reality in many projects, bearing in mind that based on our experience a UX Designer is usually a single person playing several roles.

Nevertheless, to make these roles happen, it is important that the UX Designer be a full member of the Agile team. One of the reasons is the amount of work accomplished by this role, such as: user research, market research, user-centered design, prototyping, usability inspection, user testing, visual design, coding, providing feedback and so forth.

This workload laid on the UX Designer is highlighted in Figure 11. This figure reveals how the different roles – *‘User Experience Designer’*, *‘UI Developer’* and *‘Interaction Designer’* – may be played in the different stages of an Agile cycle.

As we could notice, it is absolutely essential to spend some time before development begins on thinking holistically about the design vision.

Thus, the UX Designer cannot work on too many projects at a time. As a team member said in the study: *“UX Designers must be pigs!”*³ [C1 - PM1],

³ This is a fable told by Scrum practitioners about a pig and a chicken who considered starting a restaurant. “*We could serve ham and eggs,*” said the chicken. “*I don’t think that would work,*” said the pig. “*I’d be committed, but you’d only be involved*”.

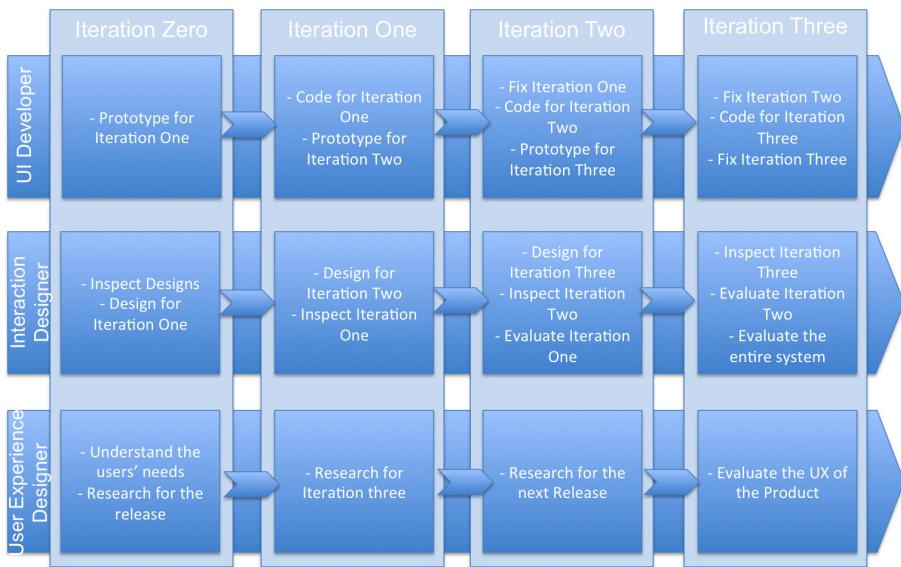


Fig. 1. UX Designer's Roles in the different Stages of an Agile cycle

expressing the importance of role accountability and involvement in projects. However, many organizations still do not consider UX Designers as full-time part of an Agile team. Thus, they keep working on too many projects at a time. This decision does not depend only on the UX Designer, but on the organizational design choices.

Notwithstanding, we should be careful on generalizing from our findings. Although the teams analyzed in these studies are considered to not be atypical, these studies do not cover all the possibilities and the contexts can vary widely.

The major contribution of this paper is to provide a better understanding of the roles played by a UX Designer within an Agile team in the different stages of Agile development.

Finally, we argue that the integration of UX Design and Agile development is a matter of culture. As UX Designers must understand the Agile culture and care because Agile adoption is on the rise and it is a completely different way of working, which requires a new approach and a new attitude toward design [4], Agile developers must understand the importance of having design in the process and how it delivers value to the product.

References

1. Hussain, Z., Slany, W., Holzinger, A.: Current state of agile user-centered design: A survey. In: Holzinger, A., Miesenberger, K. (eds.) USAB 2009. LNCS, vol. 5889, pp. 416–427. Springer, Heidelberg (2009)
2. McInerney, P., Maurer, F.: Ucd in agile projects: dream team or odd couple? *Interactions* 12, 19–23 (2005)

3. Silva da Silva, T., Silveira, M., Maurer, F., Hellmann, T.: User experience design and agile development: From theory to practice. *Journal of Software Engineering and Applications* 5(743-751) (2012)
4. Ratcliffe, L., McNeill, M.: *Agile Experience Design: A Digital Designer's Guide to Agile, Lean, and Continuous*. New Riders (2012)
5. Silva, T.S.d., Martin, A., Maurer, F., Silveira, M.: User-centered design and agile methods: A systematic review. In: Society, I.C. (ed.) *Agile Conference (Agile)*, pp. 77–86 (2011)
6. Singh, M.: U-scrum: An agile methodology for promoting usability. In: *Proceedings of the Agile 2008*, pp. 555–560. IEEE Computer Society, Washington, DC (2008)
7. Beyer, H.: *User-Centered Agile Methods*. Morgan & Claypool Publishers (2010)
8. Sy, D.: Adapting usability investigations for agile user-centered design. *Journal of Usability Studies* 2(3), 112–132 (2007)
9. Salah, D.: A framework for the integration of user centered design and agile software development processes. In: *2011 33rd International Conference on Software Engineering (ICSE)*, pp. 1132–1133. ACM (2011)
10. Ferreira, J., Sharp, H., Robinson, H.: Agile development and user experience design integration as an ongoing achievement in practice. In: *Agile 2012*, pp. 11–20 (2012)
11. Robinson, H., Segal, J., Sharp, H.: Ethnographically-informed empirical studies of software practice. *Information and Software Technology* 49(6), 540–551 (2007)
12. Lethbridge, T.C., Sim, S.E., Singer, J.: Studying software engineers: Data collection techniques for software field studies. *Empirical Soft. Eng.* (311-341) (2005)
13. Emerson, R.M., Fretz, R.I., Shaw, L.L.: *Writing Ethnographic Fieldnotes*. The University of Chicago Press (2011)

A Systematic Literature Review on Agile Development Processes and User Centred Design Integration

*

Dina Salah

Department of Computer Science
University of York, Deramore Lane
York, UK
dm560@york.ac.uk

Richard Paige

Department of Computer
Science
University of York, Deramore
Lane
York, UK
richard.paige@york.ac.uk

Paul Cairns

Department of Computer
Science
University of York, Deramore
Lane
York, UK
paul.cairns@york.ac.uk

ABSTRACT

Agile development processes and User Centred Design (UCD) integration has been gaining increased interest, in part due to the complementarity of the techniques, the benefits each can apply to the other, and the challenges associated with their combination. This paper describes a Systematic Literature Review (SLR) that was conducted on Agile and UCD integration. The aim of this SLR was to identify various challenging factors that restrict Agile and User Centred Design Integration (AUCDI) and explore the proposed practices to deal with them. The study included a total of 71 papers and excluded 80 papers published from the year 2000 till 2012. AUCDI challenges and their respective proposed practices and success factors were synthesized. A description and taxonomy of AUCDI challenges and its respective success factors and practices were reported. Practitioners can utilise the study results in identifying potential AUCDI challenges and practices or success factors to deal with them.

Agile methods are lightweight software development methods that tackle perceived limitations of plan-driven methods via a compromise between absence of a process and excessive process [29]. Agile processes aim to deal with volatile requirements via discarding upfront, precisely defined plans. They are iterative and are used to develop software incrementally. Different Agile processes implement these ideas in different ways. All Agile processes share common values and principles, defined in the *Agile Manifesto* [5].

User experience is defined as the perceptions and responses of users that result from their experience of using a product [31]. User Centred Design is a set of techniques, methods, procedures and processes as well as a philosophy that places the user at the centre of the development process [32, 19]. The goal of applying UCD is to attempt to satisfy users via producing usable and understandable products that meet their needs and interests [19].

Agile and User Centred Design Integration (AUCDI) gained increased interest due to three reasons: first, the reported advantages of UCD on the developed software as it enables developers to understand the needs of the potential users of their software, and how their goals and activities can be best supported by the software thus leading to improved usability and user satisfaction. Second, the Agile community hardly discusses users or user interfaces, thus implying either a negligence of UX or focus on less sophisticated UX projects [4]. Moreover, none of the major Agile processes explicitly include guidance for how to develop usable software [49]. In addition, the interaction design role, usability, and user interface design in an Agile team is unclear and largely overlooked [15, 6]. Furthermore, principles and practices for understanding and eliciting usability and user requirements and evaluating Agile systems for usability and UX are generally considerably deficient [49, 44, 75]. Third, there exists philosophical and principled differences between Agile methods and UCD in focus, evaluation method, culture and documentation that suggest that their integration will be fundamentally challenging.

This paper provides details of a Systematic Literature Review that was conducted on Agile and UCD integration. This SLR identified various challenging factors that restrict Agile and User Centred Design Integration (AUCDI) and

Categories and Subject Descriptors

D.2.10 [Software Engineering]: Design

General Terms

Design, Human Factors

Keywords

Agile Software Development Processes; User Centred Design; Agile User Centred Design Integration

1. INTRODUCTION

*A full version of this paper is available as *Author's Guide to Preparing ACM SIG Proceedings Using L^AT_EX2_e and BibTeX* at www.acm.org/eaddress.htm

explored the proposed practices to deal with them.

The rest of this paper is structured as follows: section 2 discusses the research method used for conducting the SLR for AUCDI. Section 3, discusses the quantitative classification results. Section 4 presents the results of the research questions in regards to AUCDI challenges, practices and success factors. Section 5 discusses the conclusion and future work.

2. SLR PROTOCOL DEVELOPMENT

This SLR had three objectives: to identify AUCDI challenges, to identify AUCDI success factors and practices and to infer relevance of AUCDI success factors and practices to AUCDI challenges.

2.1 Related Work

The AUCDI literature contains only two literature reviews that have been reported so far. The first is a literature review that discusses methods for integrating usability engineering practices into the Agile software development process and identifies the tensions between Agile methods and usability engineering [75]. The second is a SLR that revealed the existence of a common process model for integration and discussed the supporting artifacts for the collaboration between designers and developers [17]. Thus there is an absence of a SLR that provides a comprehensive scrutiny of AUCDI challenges and investigates the success factors and practices that tackle these challenges. The results of this analysis can be used by organisations that aim to achieve the integration to understand the potential challenges involved in the integration and the available practices that can be utilised to tackle these challenges.

2.2 Specifying the research question(s)

This study aim to address the following questions whose answer can play a significant role in formalizing and structuring AUCDI efforts.

- What are the challenges that could develop during AUCDI adoption process?
- What are the potential success factors for AUCDI?
- What are the potential practices for AUCDI?

2.3 Search Process

This section details the process followed to search for literature and discusses the search resources, search keywords, inclusion and exclusion criteria, data extraction strategy and data synthesis method.

2.3.1 Search Resources

The search included electronic sources, conference proceedings, journal articles and magazines. The search string focused on combining both UCD and Agile keywords and was modified in accordance to the specific search requirements of the different electronic libraries.

Electronic Sources/Digital Libraries: The electronic sources/digital libraries chosen to conduct the search on

were: The ACM Digital Library, IEEEExplore Digital Library, Google Scholar, Springer, Wiley InterScience, and Citeseer Library.

Conference Proceedings: A number of conferences proceedings were manually searched for research papers and experience reports on the topic. Those conferences included: Agile Conference, XP Conference, XP/Agile Universe, International Conference on Software Engineering (ICSE), Human Factors in Computing Systems Conference (CHI), International Symposium on Empirical Software Engineering and Measurement(ESEM), British HCI, NordicHCI, INTERACT, and European Conference on Cognitive Ergonomics.

Journals: A number of journals were manually searched including: Empirical Software Engineering, Software Practice and Experience, International Journal of Human-Computer Studies, International Journal of Human-Computer Interaction, Behavior and Information Technology, Information and Software Technology, IEEE Transactions on Software Engineering, ACM Transactions on CHI, Human Computer Interaction Journal, and Interacting with Computers.

Magazines: Three magazines were searched; IEEE Software, Communications of the ACM and Interactions. Moreover, the references of primary studies were checked for any relevant studies irrespective of the forum of publication.

2.3.2 Search Keywords

We used the research questions in order to identify the search keywords. Table 1 lists the keywords utilised.

Category	Keywords
UCD	Usability User Experience User Centred Design User Interface User Interaction Usability Engineering Human-Centred
Agile	Agile Method Agile Development Agile Practice Agile Project Scrum Extreme Programming

Table 1: Keywords for Systematic Literature Review Process

2.3.3 Study Selection Criteria

The following section discusses the criteria that were used to assess each paper and decide on whether to include or exclude primary studies.

Inclusion Criteria

To decide on paper inclusion, the following features must exist on the paper

- Peer reviewed to ensure quality of primary study.
- Available on line to ensure paper accessibility.

- Focused on the integration of UCD and Agile to ensure its relevance.
- Focused on Scrum or Extreme Programming or Agile processes in general.
- Not a workshop, panel, tutorial, seminar, interview or poster session to ensure enough details are included in the paper in order to assess its quality and use the paper in answering research questions.
- Published between the year 2000 and 2012.
- Written in English.
- Non redundant since the main focus is to study AUCDI challenges, practices and success factors. All papers written by the same authors that narrate the same AUCDI practices and success factors were excluded.

Exclusion Criteria

Any paper that does not possess any of the inclusion criteria were excluded. The remaining papers were read fully. A list of included and excluded papers was kept. Results bias was avoided via excluding multiple publications of similar research. This led in some cases to contacting authors directly to verify the most complete and recent publication. The SLR protocol was evaluated via the second and third author to check the internal consistency of the protocol in order to confirm that the research questions derive the search strings, the data extraction forms allow answering the research question(s), and the data analysis procedure is adequate to address the research questions.

2.3.4 Data Extraction and Data Synthesis Method

Data extraction forms were designed to ensure that sufficient and appropriate data is collected to address both the research questions and the quality criteria. Data extraction consistency was checked via two methods: first, data extraction was carried by the first author via randomly selecting a sample of primary studies and subjecting them to data extraction by second and third authors. The results were cross checked and any disagreements were discussed and resolved in meetings. Second, the first author conducted a test-retest process where primary studies were randomly selected and a second extraction was performed to check data extraction consistency.

The data was synthesized via thematic analysis. the iterative thematic synthesis process recommended by [16].

3. RESULTS

This section discusses both the quantitative classification results of the SLR. It starts with an overview of search sources. Then it provides an overview of the studies and discusses the excluded papers and exclusion reasons. Then it discusses the results of studies' classification.

3.1 Search Sources Overview

A number of digital libraries were searched including: Acm Digital Library, IEEEExplore Digital Library, Google Scholar, Springer, Wiley InterScience, and Citeseer Library. The

date of the search was between the period of April to September 2012 and covered the years between 2000-2012 since papers on Agile development processes started around the year 2000 whereas papers regarding the integration between Agile and UCD started shortly after that. Table 2 lists the details of the manually searched journals including the start date and start volume number and the end date and the end volume number searched.

Table 3 lists the details of the conference proceedings that were manually searched including the start and end years searched. Ten conference proceedings were manually searched.

Name of Proceedings	Start Search Year	End Search Year
Agile Conference	2003	2011
XP	2003	2012
XP-Agile Universe	2002	2004
International Conference on Software Engineering (ICSE)	2000	2011
Human Factors in Computing Systems Conference (CHI)	2000	2011
International Symposium on Empirical Software Engineering and Measurement (ESEM)	2007	2011
British HCI	2000	2010
NordicHCI	2002	2010
INTERACT	2001	2011
European Conference on Cognitive Ergonomics	2006	2011

Table 3: Conference Proceedings Searched

The earliest year for included conference proceeding is 2000 and the latest is 2012. Although the aim was to search and include all conferences from the year 2000 till the year 2012, however, some conferences have not started at the year 2000. For example, ESEM started at 2007. In addition, at the time of conducting the search some of the proceedings for the year 2012 were not available on line. Some conferences also were Biennial, for example, NordicHCI in even years and INTERACT in odd years.

3.2 Excluded Papers

The final amount of papers that were included for data analysis was 71, and a total of 80 papers were excluded. Paper exclusion was caused by a number of reasons related to format (tutorial, workshop, interview, panel, seminar), lack of peer review, lack of AUCDI focus, lack of focus on XP or Scrum, non availability on line, time constraint in case of AUCDI PhD studies, redundancy and lack of quality. A number of papers were excluded due to quality, however quality criteria were not covered due to space limitations.

3.3 Studies Classification

Table 4 shows the results of the second stage of paper classification after taking into consideration the criteria for inclusion and exclusion. A total of 80 papers were excluded for various reasons.

Studies by Year of Publication: Table 5 shows the classification of the different studies according to the publication year; 2007 and 2008 had the largest number of papers. In

Name of Journal	Start Date and volume Number for Years Searched	End Date and Volume Number for Years Searched
Empirical Software Engineering	Volume 5, Number 1, March 2000	Volume 17, Issue 4-5, August 2012
Software Practice and Experience	Volume 30, Issue 1, January 2000	Volume 42, Issue 7, July 2012
International Journal of Human-Computer Studies	Volume 52, Issue 1, January 2000	Volume 70, Issue 9, September 2012
International Journal of Human-Computer Interaction	Volume 12, Issue 1, January 2000	Volume 28, Issue 7, July 2012
Behavior and Information Technology	Volume 19, Issue 1, January 2000	Volume 31, Issue 6, June 2012
Information and Software Technology	Volume 42, Issue 1, January 2000	Volume 54, Issue 9, September 2012
IEEE Transactions on Software Engineering	Volume 26 , Issue 1, January 2000	Volume 38, Issue 3, March 2012
ACM Transactions on CHI	Volume 7 Issue 1, March 2000	Volume 19 Issue 1, March 2012
Human Computer Interaction Journal	Volume 15, Issue 1, 2000	Volume 27, Issue 1-2, April 2012
Interacting with Computers	Volume 12, Issue 3, January 2000	Volume 24, Issue 2, March 2012

Table 2: Manually Searched Journals

Exclusion Reason	No.	%
Paper Format	16	20%
Lack of Peer Review	14	17.5%
Lack of Focus on AUCDI	17	21.25 %
Lack of Focus on XP or Scrum	2	2.5%
Non Availability On line	4	5 %
Lack of Time	2	2.5 %
Redundant Content	18	22.5%
Lack of Quality	6	7.5%
Total	80	100%

Table 4: Classification Of Excluded Papers

2007 there was 11 papers whereas there was 13 papers in 2008.

Year	Papers	No.	%
2000	0	0	0%
2001	[15]	1	1.4%
2002	[65, 70]	2	2.8%
2003	[44, 36]	2	2.8%
2004	[43, 12, 4, 6, 73]	5	7%
2005	[33, 54, 59, 34, 7]	5	7%
2006	[13, 58, 51, 55]	4	5.6%
2007	[81, 20, 19, 25, 24, 23, 64, 76, 48, 56, 57]	11	15.5%
2008	[8, 74, 62, 77, 22, 30, 80, 50, 45, 63, 52, 3, 9]	13	18.3 %
2009	[1, 40, 21, 11, 18, 67, 42, 49, 72]	9	12.7 %
2010	[2, 26, 78, 46, 75, 35]	6	8.5 %
2011	[14, 69, 17, 53, 27, 47, 37, 10, 71]	9	12.7%
2012	[41, 38, 61, 28]	4	5.6%
Total		71	100%

Table 5: Studies by Year of Publication

Studies by Journal: Table 6 shows the list of different journals that included papers on AUCDI and shows that there is a scarcity in publications in journals and all jour-

nals included only one occurrence of publication on AUCDI related topics.

Name of Journal	No.	%	References
Journal of Systems and Software	1	14.3%	[41]
IFIP Advances in Information and Communication Technology	1	14.3%	[46]
Information Age	1	14.3%	[15]
Journal of Usability Studies	1	14.3%	[76]
Software Practice and Experience	1	14.3%	[27]
The Code 4 Lib Journal	1	14.3%	[52]
Cutter IT Journal	1	14.3%	[36]
Total	7	100%	-

Table 6: Studies by Journal

Studies by Conference: Table 7 shows the list of different conferences that included papers on AUCDI.

Studies by Magazine: Articles were found in two of the three searched magazines. Interactions included three articles [54, 19, 4] and Communications of the ACM included one article [14].

Studies by Book Chapter: Two book chapters were included [3, 7]

Studies by Masters/PhD: Two masters theses [69, 45] were included. However, to the best of our knowledge there is also four PhD theses that are focused on AUCDI. All of which were not included due to time constraints or lack of on line availability but papers related to these PhDs were included in the SLR.

Studies by Publication Channel and Occurrence: Table 8 shows the total amount of papers published via different publication channels. The table shows that conference papers represent the majority of publications.

Name	No.	References
Agile Conference	19	[9, 62, 44, 33, 22, 30, 58, 74, 81, 8, 23, 50, 17, 59, 42, 28, 48, 49, 10]
CHI	6	[77, 11, 80, 51, 63, 47]
XP	6	[13, 25, 2, 26, 78, 61]
HCI	3	[20, 56, 73]
XP/ Agile Universe	1	[6]
ICSE	1	[71]
GI Jahrestagung	1	[12]
OOPSLA	1	[65]
International Conference on Software Engineering Advances	1	[53]
The International Conference on Contemporary Ergonomics	1	[55]
Symposium of the Work group Human-Computer Interaction and Usability Engineering of the Austrian Computer Society on HCI and Usability for e-Inclusion	1	[40]
International Conference on Advances in Computer-Human Interactions	1	[38]
International Conference on Computer Design and Applications (ICCDA)	1	[75]
Symposium on Human Interface 2009 on Conference Universal Access in Human-Computer Interaction	1	[21]
New Zealand Computer Science Research Student Conference (NZCSRSC2007)	1	[24]
Product Focused Software Process Improvement	1	[43]
Symposium of the Workgroup Human-Computer Interaction and Usability Engineering of the Austrian Computer Society on HCI and Usability for e-Inclusion	1	[39]
HCII 2009	1	[1]
International Symposium on Intelligent Information Technology Application	1	[67]
Australian Conference on Information Systems	1	[64]
International Computer Software and Applications Conference	1	[34]
International Conference on Human-Centred Software Engineering	1	[37]
Participatory Design Conference	1	[70]
International Conference on Universal Access in Human Computer Interaction	1	[57]
International Conference on Information Technology Interfaces	1	[66]
Irish HCI	1	[72]
Total	56	-

Table 7: Studies by Conference

Publication Channel	No.	%
Conference	56	79%
Journal	7	10%
Magazine	4	6%
Masters Theses	2	3%
Book Chapter	2	3%
Total	71	100%

Table 8: Studies by Publication Channel and Occurrence

4. AUCDI CHALLENGES AND PRACTICES

This section is focused on reporting the results of the SLR research questions in regards to AUCDI challenges, practices and success factors. It reports on AUCDI challenges and the practices that have been reported in literature to tackle each challenge. These challenges fall into 3 main categories; UCD infrastructure, people, and process.

4.1 Lack of Allocated Time for Upfront Activities

Agile Methods discourages upfront planning activities since it strives to remain responsive to changing requirements [24, 49]. Moreover, Agile approaches focus on frequently producing deliverable solely in terms of functionality [51, 74]. This has resulted in lack of allocated time for software design planning activities [21, 45], performing user research to discover the problems, work practices and work flows of end users [13, 45, 21, 79, 19] and sketching out a coherent design [45, 21, 79, 51]. Moreover, incremental Agile development is translated into sliced or "feature by feature" development for design that can result in user interface that is disjoint, piecemeal and lacks a holistic, coherent, and overall structure and vision [55, 63, 2, 49, 51, 74, 60, 54, 58, 4, 45].

Practices and Success Factors: Lack of allocated time for upfront activities was addressed via upfront design. Upfront design is a separate pre-development period that is used in Agile projects for eliciting requirements, understanding users, user goals and context of use and conducting UX design up front and ahead of developers in order to achieve a comprehensive system view [13, 40, 54, 30, 42, 41, 62, 33, 19, 76, 81, 45, 59, 54, 30, 11]. Upfront design can also be used by the development and quality assurance teams to work on back end features such as selecting development environment and system platforms [62] or features with low design cost and high development cost [59]. Upfront design is also referred to as Iteration 0. Upfront design has a positive effect in mitigating poor design judgments, poor task prioritization, costly redesign problems, usability problems and inaccurate work estimates [25].

4.2 Difficulty of Modularization/ Chunking

Design chunking is breaking design into cycle sized pieces called design chunks that incrementally add elements to the overall design and design goals [76]. The incremental nature of Agile processes makes design chunking more critical and challenging [22, 76, 59]. This is due to a number of reasons: first, difficulty in determining the right chunk size and the right amount of interaction design work per iteration [78]. Second, difficulty of maintaining the ordering dependency between design chunks [76]. Third, difficulty in differentiating between user experience design activities that contribute to breadth or depth [33]. Fourth, interaction designers adopt a holistic view to interaction design and as a result it can be difficult for them to grasp and adopt design chunking both as an idea and as a work procedure [76].

Practices and Success Factors: Design chunking was addressed via a number of practices: having well defined design goals [76], using one release to chunk large or complex features [62, 21], chunking design into features [62], time boxing highly creative UX design activities [33] and postponing

depth based UX activities to occur later in the development life cycle in order to develop both the functional feature and its related UX design activities in the same iteration [33].

4.3 Optimizing the Work Dynamics Between Developers and UCD Practitioners

Agile development process changed the relationship between developers and UI designers [23] since the UI design became a team effort and required the UI designer to be on call to participate in ad hoc discussions [54]. Moreover, the highly compressed time scales and reliance on team self governance of Agile development processes required more active involvement from UX managers to ensure the regular inclusion of UX activities in team based planning and scheduling [19]. However, the Agile principle "Working software is the primary measure of progress" can pose a challenge on the integration since it can introduce competing goals between developers and UCD practitioners particularly when they work in parallel. Moreover, the Agile principle "Simplicity—the art of maximizing the amount of work not done—is essential" can represent an additional challenge to AUCDI efforts since simplicity in the user interface do not always align with simplicity in the implementation [49]. Ongoing and continuous communication need to be maintained between developers and UCD practitioners to avoid the occurrence of delays and bottle necks in the development process [24].

A number of practices were utilised in order to optimize the work dynamics between developers and UCD practitioners. These practices will be discussed in the following subsections

4.3.1 Sharing an Understanding of Users

Understanding the end user and their needs is necessary to design good UX [45]. Investing time to ensure that the entire team understands and agrees on the target audience results in ease in the collection and utilisation of customer input throughout the development process and allows the UCD practitioners to stay true to their vision and enable them to make decisions on feature sets and design trajectories [59].

4.3.2 Sharing an Understanding of Design Vision

The collaboration between software engineers and usability specialists should be supported via facilitating communication of design intent and rationale [3]. The best UX vision is useless if it is not communicated to the development team [45]. Thus UCD practitioners should effectively share the design vision via communicating it to the development team. This visibility of design vision minimizes rework and illuminates integration issues early on [79] and allow team members to develop and be cognizant of the key design goals of the system in order to ease decision making in case of competing concerns. Moreover, setting the shared design goals allows team members to have a common understanding of important aspects of the system from the customer's perspective. Prioritized goals also allow the team to prioritize fixes [49]. Earlier externalization of design vision to stakeholders is encouraged in order to make the development of usable software more effective, achieve better collaboration, and produce better software faster [57].

Practices and Success Factors: Sharing an understanding of the design vision was achieved via a number of tech-

niques including: the design studio [79], engaging developers in multiple design options [54, 59], developers taking part in UI specifications [2], sharing design artefacts and prototypes [10, 48, 21, 51, 9], and utilising information radiators [45].

4.3.3 Synchronizing Efforts of UCD Practitioners and Developers

Design drift is the occurrence of a difference between the implemented system from the initial design as a result of combining the efforts of developers and UCD practitioners. Synchronization allows the parallel usability and development efforts to proceed relatively smoothly. User interface consistency may be undermined as independently empowered teams evolve code in parallel, without coordinating their work [19]. As a result, synchronization points are needed to allow for close collaboration that will keep the information flowing between all parties involved in the project. Moreover, UX practitioners reported that the lack of communicating frequent changes caused a lot of confusion and required an immense effort from the UX team to handle frequent changes in addition to struggling to remain on track with the development team schedule [11].

Practices and Success Factors: Synchronizing the activities of UCD practitioners and developers was addressed via attendance of UX team in daily scrums [62, 11, 10], daily communication of UX designers to clarify design and inform the developers about additions or changes required for the UI [2, 59, 24], and increasing the visibility of UX team's work [47, 19, 8] via standup meetings [11, 59, 8].

4.4 Performing Usability Testing

Usability testing involves measuring typical users' performance on carefully prepared system tasks while watching and recording users performance and logging their software interactions [68, 38]. A number of sub challenges were related to usability testing within an Agile context including: method of usability testing, scheduling usability testing, accessing users for usability testing, and high cost of running usability sessions. Further details on those sub challenges is provided in the following subsections:

4.4.1 Method of Usability Testing

Agile time boxed nature poses challenges on conducting usability tests due to the difficulty of scheduling usability tests to evaluate and test prototypes and working builds with representative end users [48, 21, 22, 18, 19]. As a result, some Agile teams resolve to either peer test or do without usability thus jeopardizing the quality of design.

Practices and Success Factors: The effect of tight Agile time lines on conducting usability testing was reportedly overcome via preparation for user research [42], utilising discount usability engineering techniques including: heuristic evaluation [?, 56, 30] and RITE [21, 18], using low fidelity prototypes to conduct usability tests [30, 13, 40, 58], and conducting remote usability testing [19].

4.4.2 Scheduling Usability Testing

Scheduling interaction design evaluations with Agile development iterations was considered as a challenge due to lack of clarity in regards to timing of evaluations as part of the

iterative structure of the Agile development process [22]. Moreover, conducting usability testing at the end of the Agile development process could lead to insufficient time and resources to respond to emerging usability issues whereas, if usability tests were done early in the development process this could lead to introducing usability defects in later iterations. Moreover, if usability tests were carried out as frequently as feature acceptance tests this could lead to massive budget increases [44]. Furthermore, the code generated during sprints is often too unstable to be subjected to usability tests even if it was scheduled [19].

Practices and Success Factors: The completion of iterations and releases was perceived as valuable opportunities to frequently test the software usability and declared that usability testing fits well with acceptance testing [23]. Some researchers suggested fitting usability testing in the context of other Agile development tests, for example, acceptance testing sessions (in the case of XP) [22, 38, 12] and demonstration sessions (in the case of scrum) [44] could serve as opportunities for usability feedback on the implemented interaction design. Another suggested technique was introducing a mandatory UI reviews as a gate keeping tool, where two sign offs were set one for code and one for UI [2].

4.4.3 Accessing Users for Usability Testing

The compressed Agile time scale posed difficulties in organising access to the right people at the right time for usability testing [22, 33, 19]. This is due to the need to plan user involvement and schedule appointments with studies' subjects sufficiently in advance and thus may not fit with the Agile development schedule since it may require lead times of weeks.

Practices and Success Factors: The ability to access users for usability testing in Agile teams was maintained via planning in advance for user inclusion [45], utilising an existing user pool to act as development partners or design partners to conduct usability testing [21, 81, 1], using user recruiting firms to frequently schedule for usability sessions [42], conducting remote usability testing [33, 19], and collaborative (peer review) UI inspections [56] via designers, developers, end users, graphics designers and usability specialists.

4.4.4 Shorter Time to Iterate Design

The Agile tight time lines allow little time to integrate usability testing feedback into subsequent development cycles. The reporting period for usability testing can be too long and subsequently many changes can occur in the application and as a result many recommendations were obsolete [38]. Agile teams reported lack of time to respond to results of usability evaluations and user feedback [19, 60].

Practices and Success Factors: Shorter time to iterate design with user feedback was handled via dedicating cycles for working on user feedback and incorporating it into the development life cycle [54] and utilising the UX practitioner to act as an Agile customer in order to validate designs that are passed to developers to implement, participate in cycle planning and ensure incorporation of user feedback into the development life cycle [60].

4.5 Lack of Documentation

Agile approaches strive to achieve minimal documentation. However, documentation is crucial for estimation and implementation efforts and for properly integrating Agile and UCD. Furthermore, the lack of proper requirements documentation was reported to lead to confusion in regards to UX deliverable [11]. A variety of integration related issues need to be documented including; first, documenting design rationale in order to justify and record prior design decisions [54]. Second, recording the source of requirements whether they are customers, users, developers, usability experts or usability elicitation guidelines because this can affect the decision of creating new user stories or modifying existing ones [61]. Third, there is a need to document current designs and their expected delivery date, usability test results, high level progress for late stage design chunks, recommendations and fixes for working versions, user and task information from external users, especially from field visits, and the user interface design to be implemented [76].

Practices and Success Factors: Lack of documentation was addressed through documenting via wikis [63, 8, 76], documenting via webpages [81], use cases [19], scenarios [63], personas [45], sketches [45], wire frames [45], prototypes [45], design patterns [54, 56], information radiators [45], and tool support [61].

5. CONCLUSIONS AND FUTURE WORK

This SLR aimed to identify and classify various challenging factors that restrict AUCDI and explore the proposed practices and success factors to deal with these challenges. The SLR included a total of 71 papers and excluded 80 papers that were published from the year 2000 till 2012. The findings were quantitatively classified according to year of publication and publication channel. AUCDI challenges were explored and their respective proposed practices and success factors were synthesized and a description and taxonomy of AUCDI challenges and its respective success factors and practices were reported. Industrial practitioners can utilise the description and taxonomy of AUCDI challenges and corresponding practices and success factors in identifying potential challenges of AUCDI and practices or success factors to deal with these anticipated challenges.

To enhance the SLR findings an empirical study will be conducted that investigates current industrial practices for integrating Agile development processes and UCD in order to verify and complement the findings of the SLR. This empirical study aims to identify the common difficulties and concerns that hinder AUCDI attempts and the proposed integration methods.

6. REFERENCES

- [1] S. Adikari, C. McDonald, and J. Campbell. Little Design Up-Front: A Design Science Approach to Integrating Usability into Agile Requirements Engineering. In J. Jacko, editor, *Human-Computer Interaction. New Trends*, volume 5610 of *Lecture Notes in Computer Science*, pages 549–558. Springer Berlin / Heidelberg, 2009.
- [2] M. Albisetti. Launchpad's Quest for a Better and Agile User Interface. In *11th International Conference*

- on Agile Software Development, XP2010*, Trondheim, Norway, 2010.
- [3] S. Ambler. Tailoring Usability into Agile Software Development Projects. In E. Law, E. Hvannberg, and G. Cockton, editors, *Maturing Usability*, Human–Computer Interaction Series, pages 75–95. Springer London, 2008.
 - [4] J. Armitage. Are Agile Methods Good for Design? *Interactions*, 11(1):14–23, Jan. 2004.
 - [5] K. Beck. Manifesto for Agile Software Development, 2000.
 - [6] H. Beyer, K. Holtzblatt, and L. Baker. An Agile Customer-Centered Method: Rapid Contextual Design. In *XP/AU*, 2004.
 - [7] S. Blomkvist. Towards a Model for Bridging Agile Development and User-Centered Design. In A. Seffah, J. Gulliksen, and M. Desmarais, editors, *Human-Centered Software Engineering – Integrating Usability in the Software Development Lifecycle*, volume 8 of *Human–Computer Interaction Series*, pages 219–244. Springer Netherlands, 2005.
 - [8] D. Broschinsky and L. Baker. Using Persona with XP at LANDesk Software, an Avocent Company. In *Proceedings of the Agile 2008*, AGILE ’08, pages 543–548, Washington, DC, USA, 2008. IEEE Computer Society.
 - [9] J. Brown, G. Lindgaard, and R. Biddle. Stories, Sketches, and Lists: Developers and Interaction Designers Interacting Through Artefacts. In *Proceedings of the Agile 2008*, AGILE ’08, pages 39–50, Washington, DC, USA, 2008. IEEE Computer Society.
 - [10] J. Brown, G. Lindgaard, and R. Biddle. Collaborative Events and Shared Artefacts: Agile Interaction Designers and Developers Working Toward Common Aims. In *Agile Conference (AGILE)*, 2011, pages 87–96, aug. 2011.
 - [11] M. Budwig, S. Jeong, and K. Kelkar. When User Experience Met Agile: A Case Study. In *Proceedings of the 27th International Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA ’09, pages 3075–3084, New York, NY, USA, 2009. ACM.
 - [12] R. Carbon, J. Dorr, and M. Trapp. Focusing Extreme Programming on Usability. In *GI Jahrestagung*, pages 147–152, 2004.
 - [13] S. Chamberlain, H. Sharp, and N. Maiden. Towards a Framework for Integrating agile Development and User-Centred Design. In *Proceedings of the 7th International Conference on Extreme Programming and Agile Processes in Software Engineering*, XP’06, pages 143–153, Berlin, Heidelberg, 2006. Springer-Verlag.
 - [14] T. Coatta and R. Rutter. UX Design and Agile: A Natural Fit? *Communications of the ACM*, 54(1):54–60, 2011.
 - [15] L. Constantine. Process Agility and Software Usability: Towards Lightweight Usage Centred Design, 2001.
 - [16] D. Cruzes and T. Dyba. Recommended Steps for Thematic Synthesis in Software Engineering. In *Proceedings of the 2011 International Symposium on Empirical Software Engineering and Measurement*, ESEM ’11, pages 275–284, Washington, DC, USA, 2011. IEEE Computer Society.
 - [17] T. S. da Silva, A. Martin, F. Maurer, and M. Silveira. User-Centered Design and Agile Methods: A Systematic Review. In *Agile Conference (AGILE)*, 2011, pages 77–86, aug. 2011.
 - [18] D. Dayton and C. Barnum. The Impact of Agile on User-Centered Design: Two Surveys Tell the Story. *Technical Communication*, 56(3):219–234, 2009.
 - [19] M. Detweiler. Managing UCD within Agile Projects. *Interactions*, 14(3):40–42, May 2007.
 - [20] M. Duchtig, D. Zimmermann, and K. Nebe. Incorporating User Centered Requirement Engineering into Agile Software Development. In *Proceedings of the 12th international Conference on Human-Computer Interaction: Interaction Design and Usability*, HCI’07, pages 58–67, Berlin, Heidelberg, 2007. Springer-Verlag.
 - [21] M. Federoff and C. Courage. Successful User Experience in an Agile Enterprise Environment. In *Proceedings of the Symposium on Human Interface 2009 on Conference Universal Access in Human-Computer Interaction. Part I: Held as Part of HCI International 2009*, pages 233–242, Berlin, Heidelberg, 2009. Springer-Verlag.
 - [22] J. Ferreira. Interaction Design and Agile Development: Reconciling Iterative and Incremental Approaches. In *Agile Conference*, 2008.
 - [23] J. Ferreira, J. Noble, and R. Biddle. Agile Development Iterations and UI Design. In *Proceedings of the AGILE 2007*, AGILE ’07, pages 50–58, Washington, DC, USA, 2007. IEEE Computer Society.
 - [24] J. Ferreira, J. Noble, and R. Biddle. Interaction Designers on eXtreme Programming Teams: Two Case Studies from the Real World. In *Proceedings of the Fifth New Zealand Computer Science Research Student Conference (NZCSRSC2007)*, 2007.
 - [25] J. Ferreira, J. Noble, and R. Biddle. Up-front Interaction Design in Agile Development. In *Proceedings of the 8th International Conference on Agile Processes in Software Engineering and Extreme Programming*, XP’07, pages 9–16, Berlin, Heidelberg, 2007. Springer-Verlag.
 - [26] J. Ferreira, H. Sharp, and H. Robinson. Values and Assumptions Shaping Agile Development and User Experience Design in Practice. In A. Sillitti, A. Martin, X. Wang, E. Whitworth, W. Aalst, J. Mylopoulos, M. Rosemann, M. J. Shaw, and C. Szyperski, editors, *Agile Processes in Software Engineering and Extreme Programming*, volume 48 of *Lecture Notes in Business Information Processing*, pages 178–183. Springer Berlin Heidelberg, 2010.
 - [27] J. Ferreira, H. Sharp, and H. Robinson. User Experience Design and Agile Development: Managing Cooperation through Articulation Work. *Software Practice and Experience*, 41(9):963–974, Aug. 2011.
 - [28] J. Ferreira, H. Sharp, and H. Robinson. Agile Development and User Experience Design Integration as an On-going Achievement in Practice. In *Agile 2012*, 2012.
 - [29] M. Fowler. The New Methodology, December 2005.
 - [30] D. Fox, J. Sillito, and F. Maurer. Agile Methods and

- User-Centered Design: How These Two Methodologies are Being Successfully Integrated in Industry. In *Proceedings of the Agile 2008*, AGILE '08, pages 63–72, Washington, DC, USA, 2008. IEEE Computer Society.
- [31] J.-J. Garrett. *The Elements of User Experience: User Centered Design for the Web and Beyond*. New Riders, 2011.
 - [32] J. Gould and C. Lewis. Designing for Usability: Key Principles and What Designers Think. *Communications of ACM*, 28(3):300–311, Mar. 1985.
 - [33] P. Hodgetts. Experiences Integrating Sophisticated User Experience Design Practices into Agile Processes. In *Proceedings of the Agile Development Conference*, ADC '05, pages 235–242, Washington, DC, USA, 2005. IEEE Computer Society.
 - [34] A. Holzinger, M. Errath, G. Searle, B. Thurnher, and W. Slany. From Extreme Programming and Usability Engineering to Extreme Usability in Software Engineering Education (XP+UE→XU). In *Proceedings of the 29th Annual International Conference on Computer Software and Applications Conference*, COMPSAC-W'05, pages 169–172, Washington, DC, USA, 2005. IEEE Computer Society.
 - [35] A. Hosseini-Khayat, T. Hellmann, and F. Maurer. Distributed and Automated Usability Testing of Low-Fidelity Prototypes. In *Agile Conference (AGILE)*, pages 59 –66, aug. 2010.
 - [36] W. Hudson. Adopting User-Centered Design within an Agile Process: A Conversation. *Cutter IT Journal*, 16:10:5–12, 2003.
 - [37] S. R. Humayoun, Y. Dubinsky, and T. Catarci. A Three-Fold Integration Framework to Incorporate User-Centered Design into Agile Software Development. In *Proceedings of the 2nd International Conference on Human Centered Design*, HCD'11, pages 55–64, Berlin, Heidelberg, 2011. Springer-Verlag.
 - [38] Z. Hussain, H. Milchrahm, S. Shahzad, W. Slany, M. Umgeher, T. Vlk, C. Koefel, M. Tscheilgi, and P. Wolkerstorfer. Practical Usability in XP Software Development Processes. In *The Fifth International Conference on Advances in computer Human Interactions, ACHI 2012*, 2012.
 - [39] Z. Hussain, W. Slany, and A. Holzinger. Current State of Agile User-Centered Design: A Survey. In *Proceedings of the 5th Symposium of the Workgroup Human-Computer Interaction and Usability Engineering of the Austrian Computer Society on HCI and Usability for e-Inclusion*, USAB '09, pages 416–427, Berlin, Heidelberg, 2009. Springer-Verlag.
 - [40] Z. Hussain, W. Slany, and A. Holzinger. Investigating Agile User-Centered Design in Practice: A Grounded Theory Perspective. In *Proceedings of the 5th Symposium of the Workgroup Human-Computer Interaction and Usability Engineering of the Austrian Computer Society on HCI and Usability for e-Inclusion*, USAB '09, pages 279–289, Berlin, Heidelberg, 2009. Springer-Verlag.
 - [41] Z. Hussain, W. Slany, and A. Holzinger. Agile Software Developement Methods and Usability/ User Centred Design: Prospectives from an Online Survey. *Journal of Systems and Software*, 2012.
 - [42] T. Illmensee and A. Muff. 5 Users Every Friday: A Case Study in Applied Research. In *Proceedings of the 2009 Agile Conference*, AGILE '09, pages 404–409, Washington, DC, USA, 2009. IEEE Computer Society.
 - [43] T. Jokela and P. Abrahamsson. Usability Assessment of an Extreme Programming Project: Close Co-operation with the Customer Does Not Equal to Good Usability. In F. Bomarius and H. Iida, editors, *Product Focused Software Process Improvement*, volume 3009 of *Lecture Notes in Computer Science*, pages 393–407. Springer Berlin / Heidelberg, 2004.
 - [44] D. Kane. Finding a Place for Discount Usability Engineering in Agile Development: Throwing Down the Gauntlet. In *Proceedings of the Conference on Agile Development*, ADC '03, Washington, DC, USA, 2003. IEEE Computer Society.
 - [45] J. Kollmann. Designing the User Experience in an Agile Context. Master's thesis, University College London, 2008.
 - [46] M. Larusdottir, E. Bjarnadottir, and J. Gulliksen. The Focus on Usability in Testing Practices in Industry. In P. Forbrig, F. Patern?, and A. Mark Pejtersen, editors, *Human-Computer Interaction*, volume 332 of *IFIP Advances in Information and Communication Technology*, pages 98–109. Springer Boston, 2010.
 - [47] J. C. Lee, T. Judge, and S. McCrickard. Evaluating EXtreme Scenario-Based Design in a Distributed Agile Team. In *Proceedings of the 2011 Annual Conference on Human factors in Computing Systems*, CHI EA '11, pages 863–877, New York, NY, USA, 2011. ACM.
 - [48] J. C. Lee and S. McCrickard. Towards Extreme(ly) Usable Software: Exploring Tensions Between Usability and Agile Software Development. In *Proceedings of the AGILE 2007*, AGILE '07, pages 59–71, Washington, DC, USA, 2007. IEEE Computer Society.
 - [49] J. C. Lee, S. McCrickard, and T. Stevens. Examining the Foundations of Agile Usability with eXtreme Scenario-Based Design. In *Agile Conference*, pages 3 –10, aug. 2009.
 - [50] A. Leszek and C. Courage. The Doctor is "In" – Using the Office Hours Concept to Make Limited Resources Most Effective. In *AGILE Conference 2008*, pages 196 –201, aug. 2008.
 - [51] M. Lievesley and J. Yee. The Role of the Interaction Designer in an Agile Software Development Process. In *CHI '06 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '06, pages 1025–1030, New York, NY, USA, 2006. ACM.
 - [52] H. Lindstrom and M. Malmsten. User-Centred Design and Agile Development: Rebuilding the Swedish National Union Catalogue. *The Code4Lib Journal*, 5:12–Ü15, 2008.
 - [53] B. Losada, M. Urretavizcaya, and I. Fernandez-deCastro. Agile Development of Interactive Software by means of User Objectives. In *The Sixth International Conference on Software Engineering Advances*, 2011.
 - [54] P. McInerney and F. Maurer. UCD in Agile Projects: Dream Team or Odd Couple? *Interactions*, 12(6):19–23, Nov. 2005.
 - [55] M. Mcneil. Agile User-Centred Design. In *Proceedings*

- of the International Conference on Contemporary Ergonomics (CE2006)*, Cambridge, UK, 2006.
- [56] T. Memmel, F. Gundelsweiler, and H. Reiterer. Agile Human-Centred Software Engineering. In *Proceedings of the 21st British HCI Group Annual Conference on People and Computers: HCI...but not as we know it - Volume 1*, BCS-HCI '07, pages 167–175, Swinton, UK, UK, 2007. British Computer Society.
- [57] T. Memmel, H. Reiterer, and A. Holzinger. Agile Methods and Visual Specification in Software Development: A Chance to Ensure Universal Access. In *Proceedings of the 4th International Conference on Universal Access in Human Computer Interaction: Coping with Diversity*, UAHCI'07, pages 453–462, Berlin, Heidelberg, 2007. Springer-Verlag.
- [58] G. Meszaros and J. Aston. Adding Usability Testing to an Agile Project. In *Proceedings of the conference on AGILE 2006*, AGILE '06, pages 289–294, Washington, DC, USA, 2006. IEEE Computer Society.
- [59] L. Miller. Case Study of Customer Input For a Successful Product. In *Proceedings of the Agile Development Conference*, ADC '05, pages 225–234, Washington, DC, USA, 2005. IEEE Computer Society.
- [60] L. Miller and D. Sy. Agile User Experience SIG. In *Proceedings of the 27th International Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '09, pages 2751–2754, New York, NY, USA, 2009. ACM.
- [61] A. Moreno and A. Yagie. Agile User Stories Enriched with Usability. In C. Wohlin, editor, *Agile Processes in Software Engineering and Extreme Programming*, volume 111 of *Lecture Notes in Business Information Processing*, pages 168–176. Springer Berlin Heidelberg, 2012.
- [62] M. Najafi and L. Toyoshiba. Two Case Studies of User Experience Design and Agile Development. In *Proceedings of the Agile 2008*, AGILE '08, pages 531–536, Washington, DC, USA, 2008. IEEE Computer Society.
- [63] H. Obendorf and M. Finck. Scenario-Based Usability Engineering Techniques in Agile Development Processes. In *CHI '08 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '08, pages 2159–2166, New York, NY, USA, 2008. ACM.
- [64] D. Parsons, R. Lal, and H. Ryu. Software Developement Methodologies, Agile Developement and Usability Engineering. In *18th Australian Conference on Information Systems*, 2007.
- [65] J. Patton. Hitting the Target: Adding Interaction Design to Agile Software Development. In *OOPSLA 2002 Practitioners Reports*, OOPSLA '02, New York, NY, USA, 2002. ACM.
- [66] C. S. A. Peixoto. Human-Computer Interface Expert System for Agile Methods. In *International Conference on Information Technology Interfaces*, pages 311–316, june 2009.
- [67] C. S. A. Peixoto and A. E. A. da Silva. A Conceptual Knowledge Base Representation for Agile Design of Human-Computer Interface. In *Proceedings of the 3rd international Conference on Intelligent Information Technology Application*, IITA'09, pages 156–160, Piscataway, NJ, USA, 2009. IEEE Press.
- [68] J. Preece, Y. Rogers, and H. Sharp. *Interaction Design: Beyond Human Computer Interaction*. John Wiley & Sons, 2002.
- [69] P. Rannikko. User Centred Design in Agile Software Developement. Master's thesis, University of Tampere, School of Information Science, April 2011.
- [70] M. Rittenbruch, G. McEwan, nigel Ward, T. Mansfield, and D. Bartenstein. Extreme Participation - Moving Extreme Programming Towards Participatory Design. In *Proceedings of the Seventh Biennial Participatory Design Conference*, 2002.
- [71] D. Salah. A Framework for the Integration of User Centered Design and Agile Software Development Processes. In *33rd International Conference on Software Engineering (ICSE)*, pages 1132–1133, may 2011.
- [72] D. Salah, H. Petrie, and R. Paige. Towards a Framework for Bridging User-Centred Design and Agile Software Development Processes. In *3rd Irish HCI Conference 2009*, 2009.
- [73] H. Sharp, H. Robinson, and J. Segal. Integrating User-Centred Design and Software Engineering: a Role for eXtreme Programming? In *BCS-HCI Group's 7th Educators Workshop: Effective Teaching and Training in HCI*, 2004.
- [74] M. Singh. U-SCRUM: An Agile Methodology for Promoting Usability. In *Agile Conference*, pages 555 –560, aug. 2008.
- [75] O. Sohaib and K. Khan. Integrating Usability Engineering and Agile Software Development: A Literature Review. In *Computer Design and Applications (ICCDA), 2010 International Conference on*, volume 2, pages V2–32 –V2–38, june 2010.
- [76] D. Sy. Adapting Usability Investigations for Agile User-Centred Design. *Journal of Usability Studies*, 2(3):112–132, May 2007.
- [77] D. Sy and L. Miller. Optimizing Agile User-Centred Design. In *CHI '08 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '08, pages 3897–3900, New York, NY, USA, 2008. ACM.
- [78] K. Tzanidou and J. Ferreira. Design and Development in the \$Agile Room\$: Trialing Scrum at a Digital Agency. In A. Sillitti, A. Martin, X. Wang, and E. Whitworth, editors, *Agile Processes in Software Engineering and Extreme Programming*, volume 48 of *Lecture Notes in Business Information Processing*, pages 372–378. Springer Berlin Heidelberg, 2010.
- [79] J. Ungar. The Design Studio: Interface Design for Agile Teams. In *Proceedings of the Agile 2008*, AGILE '08, pages 519–524, Washington, DC, USA, 2008. IEEE Computer Society.
- [80] J. Ungar and J. White. Agile User Centered Design: Enter the Design Studio - A Case Study. In *CHI '08 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '08, pages 2167–2178, New York, NY, USA, 2008. ACM.
- [81] H. Williams and A. Ferguson. The UCD Perspective: Before and After Agile. In *Proceedings of the AGILE 2007*, AGILE '07, pages 285–290, Washington, DC, USA, 2007. IEEE Computer Society.



The Usability Metric for User Experience

Kraig Finstad

Intel® Corporation, 2501 NW 229th Ave., M/S RA1-222, Hillsboro, OR 97124, United States

ARTICLE INFO

Article history:

Received 21 September 2009

Accepted 6 April 2010

Available online 6 May 2010

Keywords:

Usability

User experience

Scale

Metric

ABSTRACT

The Usability Metric for User Experience (UMUX) is a four-item Likert scale used for the subjective assessment of an application's perceived usability. It is designed to provide results similar to those obtained with the 10-item System Usability Scale, and is organized around the ISO 9241-11 definition of usability. A pilot version was assembled from candidate items, which was then tested alongside the System Usability Scale during usability testing. It was shown that the two scales correlate well, are reliable, and both align on one underlying usability factor. In addition, the Usability Metric for User Experience is compact enough to serve as a usability module in a broader user experience metric.

© 2010 Elsevier B.V. All rights reserved.

1. Introduction

Measuring and tracking usability is an ongoing challenge for organizations that are concerned with improving user experience. A popular and cost-effective approach to usability measurement is the use of standardized surveys. When the Information Technology (IT) department at Intel® decided to standardize on a usability inventory, it selected the System Usability Scale (SUS). The SUS is a 10-item, five-point Likert scale with a weighted scoring range of 0–100 and which has been shown to be a reliable measure of usability. It is anchored with one as Strongly Disagree and five as Strongly Agree. According to [Holyer \(1993\)](#), it correlates at 0.86 with the 50-item Software Usability Measurement Inventory ([Kirkowski et al., 1992](#)). [Tullis and Stetson \(2004\)](#) found the SUS to outperform the Questionnaire for User Interface Satisfaction ([Chin et al., 1988](#)) and the Computer System Usability Questionnaire ([Lewis, 1995](#)) at assessing website usability. The SUS was adopted as a standard usability measure because of these performance characteristics, in addition to being free and relatively compact. It proved to be easy for project teams to understand, but several issues emerged. As IT at Intel® began to pursue a more comprehensive approach to user experience, the SUS was originally considered as a usability module for a more comprehensive index of user experience. This definition describes user experience as a lifecycle consisting of: Marketing and Brand Awareness, Acquisition and Installation, Product or Service Use, Product Support, and Removal/End of Life ([Sward and Macarthur, 2007](#)). However, it became apparent that simply adapting the SUS to work as a Product Use component was not feasible. Early trials with internal project teams showed that a 10-item Product Use module would

be too large when other elements such as Product Support were factored in and required their own additional scales. The concept of user experience covers a lot of ground: any Product Use or usability component of a larger user experience index would have to be much more compact than 10 items. Also, in its original form, the SUS did not lend itself well to electronic distribution in a global environment due to non-native English speakers not understanding the word "cumbersome" in SUS Item 8 ([Finstad, 2006](#)), and it used a five-point Likert scale which has been shown to be inadequate in many cases. [Diefenbach et al. \(1993\)](#) found that seven-point scales outperformed five-point scales in reliability, accuracy, and ease of use, while [Cox's \(1980\)](#) review of Likert scales found the optimal number of alternatives to be seven. [Finstad \(in press\)](#) found that respondents were more likely to provide non-integer interpolations (e.g., saying "three and a half" instead of "three" or "four") in the five-point SUS than in a seven-point alternate version of the same instrument. These interpolations indicate a mismatch between the scale and a user's actual evaluation. From a more theoretical standpoint, the SUS items did not map well onto the concepts that comprise usability according to [ISO 9241-11 \(1998\)](#), namely effectiveness, efficiency, and satisfaction. These mappings are important because the SUS is not a diagnostic tool; it can indicate whether there is a problem with a system's usability but not what those problems actually are. It is often used as a starting point in usability efforts, but an alignment with known usability factors can provide a stronger foundation for user experience efforts.

These issues with the SUS motivated a research program aimed at developing a replacement. The goal was to provide an inventory that was substantially shorter than the SUS and therefore appropriate as the usability component of a larger user experience index. An early attempt at item set reduction aimed to leverage a single

E-mail address: kraig.a.finstad@intel.com

ease of use item from the SUS. A SUS survey with 43 responses was conducted on an enterprise portal product. It was found that Item 3 in the SUS, "I thought [the system] was easy to use" correlated with the final SUS score at $r = 0.89$, $p < 0.01$; the strongest correlation in the set of 10 items. This result was not surprising in light of recent findings. **Sauro and Dumas (2009)** have demonstrated the utility of a general ease of use Likert item, and have also shown a promising alternative in the Subjective Mental Effort Questionnaire (SMEQ). **Tedesco and Tullis (2006)** found that a single "Overall this task was: Very Difficult...Very Easy" Likert item correlated significantly with usability test performance. This direction motivated further analysis of SUS surveys and SUS Item 3 with other systems, but no consistent pattern emerged. In some cases SUS Item 3 correlated most strongly with the final SUS score, and in others it did not. The idea of reducing an instrument to one general ease of use item was abandoned. Instead, a new direction was taken – the development of a concise scale that would more closely conform to the ISO 9241-11 (1998) definition of usability, would minimize bias and language issues, and would still perform as well as the baseline it was intended to replace. In this case the baseline was the updated, internationally-appropriate SUS (with "cumbersome" clarified as "awkward") and the performance goal of total SUS score to the total score of the new scale was set at a correlation of 0.80 or better. The resulting instrument is the Usability Metric for User Experience (UMUX), and this paper outlines the research and development of this usability component of a more general user experience measurement model.

2. Pilot study

A pilot study was developed to explore these possibilities. The end goal of the pilot study was the determination of how candidate Likert items would fare in an analysis of actual responses to items.

2.1. Method

2.1.1. Participants

A total of 42 Intel® employees were recruited as part of a larger worldwide usability test. As a control for cultural and language factors in both the usability task and the candidate Likert items, participants were recruited worldwide. Users from the United States, Germany, Ireland, the Netherlands, China, the Philippines, Malaysia, and Israel participated in this study.

2.1.2. Materials

A pool of candidate Likert items was developed that were related to the ISO 9241-11 (1998) definition of usability. A total of 12 such items were developed, four each for effectiveness, efficiency, and satisfaction. Some were intentionally generic, while others were behavior-based (e.g., "I don't make many errors with this system") or emotion-based (e.g., "I would prefer to use something other than this system"). These candidate items used a five-point scale so they could be used alongside the SUS in an actual post-deployment usability survey. Also like the SUS, they used an alternating positive/negative keying to control for acquiescence bias. These 12 candidate items and their usability factors are listed in Table 1.

2.1.3. Design and procedure

Participants first engaged in a usability test of an enterprise software prototype involving the selection of contract workers and adding them to a database. After completing the usability test, participants received a modified version of the SUS. The first three items were candidate items, followed by the SUS, which was then followed by three more candidate items. This format presented the

SUS as an intact instrument in order to achieve a valid final score for analysis. Each participant therefore responded to six candidate items, two per usability component (effectiveness, efficiency, and satisfaction), in addition to the SUS. This allowed a direct per-participant comparison of candidate item responses with a final SUS score. Presentation of candidate items was counterbalanced across participants. Response to the Likert items was verbal, with the entire items read aloud to help ensure comprehension of the scale. The facilitator recorded responses manually. After completion of the composite survey, participants were thanked for their time, debriefed, and excused.

2.2. Results

2.2.1. Item correlations

The odd items in the SUS were scored as [score - 1], and the even items were scored as [5 - score]. This aligned all scores in one direction, removing the positive/negative keying of the language in the instrument. It also allowed zeroes at the bottom of the range. The ten rescored items were summed and multiplied by 2.5, providing a range of 0–100 (Brooke, 1996). The critical measure of this study was the correlation of UMUX candidate items (scored similarly to the SUS) with the final SUS score. A high correlation coefficient indicated that the candidate item was in line with the total SUS score, regardless of direction. That is, a good candidate item would correlate highly with the SUS regardless of whether the SUS itself was indicating good or poor usability. This is a different approach from that used in developing the original SUS, which selected candidates based on their tendencies toward extreme (non-neutral) responses (Brooke, 1996). The UMUX is intended to match the performance of the SUS, so alignment with existing measures is more important.

As the UMUX was being designed to reflect the ISO 9241-11 (1998) definition of usability with as few items as possible, the highest-correlating candidate items for each usability component were chosen for further study. Table 2 below summarizes these results.

All the correlations in this table were negative due to the negative keying of the candidates; for instance, if the application was usable then the participants disagreed on the item. The more general items with language like "I am satisfied..." tended to correlate poorly. As a point of comparison, the correlations of the items in the SUS to the SUS score itself varied from $r = 0.36$ to $r = 0.78$.

No participants required assistance with the terminology or phrasing of the UMUX candidate items. This was taken as evidence

Table 1
Candidate items used (pilot study).

Usability component	Candidate item
Efficiency	[This system] saves me time. I tend to make a lot of mistakes with [this system]. I don't make many errors with [this system]. I have to spend a lot of time correcting things with [this system].
Effectiveness	[This system] allows me to accomplish my tasks. I think I would need a system with more features for my tasks. I would not need to supplement [this system] with an additional one. [This system's] capabilities would not meet my requirements.
Satisfaction	I am satisfied with [this system]. I would prefer to use something other than [this system]. Given a choice, I would choose [this system] over others. Using [this system] was a frustrating experience.

Note: Bracketed text is custom-replaced by relevant system.

Table 2

Items having highest correlation with overall SUS score (pilot study).

Usability component	Candidate UMUX Item	r
Efficiency	I have to spend a lot of time correcting things with [this system].	-0.48*
Effectiveness	[This system's] capabilities would not meet my requirements.	-0.50*
Satisfaction	Using [this system] was a frustrating experience.	-0.76*

Note: Bracketed text is custom-replaced by relevant system.

* $p < 0.05$.

that the items were appropriate for an international English-speaking audience.

2.2.2. Analysis of preliminary instrument

These results motivated an analysis to determine how the best candidate items would perform if they comprised an actual instrument that yielded a SUS-like usability score. If candidate item data from the pilot study could produce a result comparable to the SUS, those items would be subjected to a wider scale validation study with new participants. The preliminary UMUX was comprised of the three highest-correlating candidate items from each ISO (1998) usability factor shown in Table 2, plus the overall ease of use from the SUS ("I thought the system was easy to use"), which had shown earlier to be promising as a general question with $r = 0.89$, $p < 0.01$ (see Section 1).

Data for the analysis consisted of the 21 response sets from the pilot study that included the candidate items. These four candidate items from a five-point scale were used with a 2.5 multiplier, providing a score range of 0–40 (compared to 0–100 for the SUS). The preliminary UMUX attained a mean score of 24 out of 40, and the SUS for the same participants attained a mean score of 60 out of 100. Both of these scores were 60% of their respective maximums. The preliminary UMUX correlated with the SUS at $r = 0.81$, $p < 0.1$.

2.3. Discussion

The pilot study identified the three most promising candidates to be included in a measurement instrument along with an additional ease of use item. The results for the candidate items were in line with correlations achieved by the SUS itself. When combined into a preliminary user experience inventory, the four candidate items met the research program's goal of a correlation higher than 0.80 with the SUS.

3. Survey study

The next step was to design an experiment directly comparing the SUS with the new UMUX instrument.

3.1. Method

3.1.1. Participants

Participants consisted of two groups of users of enterprise software at Intel®. System 1 was a contract worker enterprise application that had been rated as having poor usability, and System 2 was an audio-conferencing application that had been rated as having good usability. Valid responses received from survey requests resulted in System 1 with $n = 273$ and System 2 with $n = 285$.

3.1.2. Materials

Some minor changes were made to the candidate items to build the experimental UMUX to better balance the positive/negative keying and to clear up some potential confusion with the Efficiency item. For comparison with the original items in Table 2, see the completed UMUX in Table 3.

Table 3

Usability components and scale items (survey study).

Usability component	Candidate UMUX item
Effectiveness	[This system's] capabilities meet my requirements.
Satisfaction	Using [this system] is a frustrating experience.
Overall	[This system] is easy to use.
Efficiency	I have to spend too much time correcting things with [this system].

Note: Bracketed text is custom-replaced by relevant system.

The UMUX used in this survey study was a seven-point Likert scale, anchored with one as Strongly Disagree and seven as Strongly Agree. Like the SUS, all other response options were numbered but otherwise unlabeled. This move to a seven-point scale gave the UMUX an initial range of 0–60, after applying the 2.5 multiplier from the SUS. These UMUX scores could be presented as a percentage of the maximum (60) to provide a final range comparable to that found in the SUS (0–100). The SUS was also modified as per Finstad (2006), clarifying "cumbersome" as "awkward".

3.1.3. Design and procedure

This study used a between-subjects design, with participants using one of two systems (having poor usability or good usability) and then responding to both the UMUX and the SUS. Presentation of the instruments was counterbalanced so that half the participants responded to the UMUX first, and the other half responded to the SUS first. These instruments were administered electronically through a combination of email invitation and online survey tool.

3.2. Results

3.2.1. Principal components

A common first step in validating instruments is through principal components analysis (Tabachnik and Fidell, 1989). The results from the initial principal component extraction are shown below in Table 4.

The strength of the first principal component led to the conclusion that UMUX items were aligning along one usability component. This perspective is supported by the scree plot of the components, shown in Fig. 1 below.

Tabachnik and Fidell (1989) recommend the point where the scree plot line changes direction as a determinant of the number of components; this plot's direction drops off dramatically after the first component. This is strong evidence for the scale measuring one "usability" component. Because no secondary components

Table 4

Principal components (survey study).

Principal component	Eigenvalue	Percent of variance explained
1	3.37	84.37
2	0.31	7.83
3	0.20	4.88
4	0.12	2.92

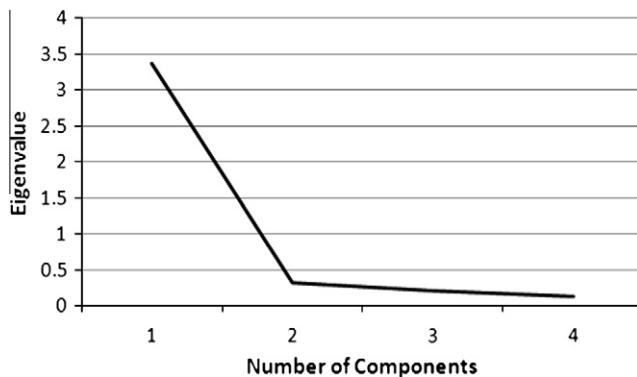


Fig. 1. Scree plot of principal components (survey study).

emerged from the analysis, no attempts at further extractions or rotations were performed. The SUS provided a similar one-component extraction, with no additional elements emerging. For a more thorough treatment of factoring in the SUS, see Lewis and Sauro (2009), who found evidence that the SUS may be comprised of two factors (usability and learnability). The conclusion from this analysis is that both instruments were unidimensional and align on just one component (usability) rather than several.

3.2.2. Reliability

Instruments need to measure an underlying construct consistently. At the early stages of a metric's development, one way to establish this is through reliability estimation. Cronbach's alpha is a correlation coefficient that indicates how well a factor is being measured. The rule of thumb for Cronbach's alpha is that a coefficient of higher than an absolute value of 0.70 indicates a high degree of internal reliability. Instruments farther along in their development are subjected to more longitudinal reliability measures. The Cronbach's alpha for both instruments indicated high reliability: 0.94 for the UMUX and 0.97 for the SUS. Therefore, both instruments were reliable.

3.2.3. Validity and sensitivity

The overall correlation of UMUX with the SUS, across both system conditions, was $r = 0.96$, $p < 0.001$. These results exceed the goal criterion of $r > 0.80$, providing evidence of validity. *T*-tests demonstrated that System 2 was more usable than System 1, $t(533) = -39.04$, $r = 0.86$, $p < 0.01$ for UMUX, $t(556) = -44.47$, $r = 0.89$, $p < 0.01$ for SUS, thereby providing evidence for sensitivity. The breakdown of usability inventory scores and correlations is shown in Table 5.

3.2.4. Item correlations

After the UMUX had been developed and finalized, the performance of its individual items was examined in two applied situations. All final UMUX items were analyzed for their contribution to the overall UMUX score, both as a post-usability test questionnaire ($n = 45$) and in the first seven internal usability projects completed with the new scale as a standard instrument ($n = 272$). The results shown in Table 6 demonstrate significant item-total corre-

Table 5

Means, standard deviations, and correlation (survey study).

System	UMUX (0–100)	SUS (0–100)	<i>r</i>
System 1	27.66 (20.54)	28.77 (18.19)	0.84*
System 2	87.91 (15.98)	88.39 (13.18)	0.81*

* $p < 0.001$.

Table 6
Correlations of UMUX items with overall score (survey study).

Scale item	Post-test <i>r</i>	Surveys <i>r</i>
1. [This system's] capabilities meet my requirements.	0.78*	0.85*
2. Using [this system] is a frustrating experience.	-0.76*	-0.89*
3. [This system] is easy to use.	0.76*	0.87*
4. I have to spend too much time correcting things with [this system].	-0.69*	-0.81*

* $p < 0.05$.

lations. It was therefore concluded that all UMUX items were valid contributors to the overall score.

4. Discussion

4.1. Implementation

The UMUX can be administered electronically as a survey, or as a follow-up in usability testing. It is simple to administer, as it requires no branching or reordering of items. The UMUX is implemented as shown below, where bracketed text is custom-replaced by the relevant system.

1. [This system's] capabilities meet my requirements.	1 2 3 4 5 6 7	Strongly Disagree	Strongly Agree
2. Using [this system] is a frustrating experience.	1 2 3 4 5 6 7	Strongly Disagree	Strongly Agree
3. [This system] is easy to use.	1 2 3 4 5 6 7	Strongly Disagree	Strongly Agree
4. I have to spend too much time correcting things with [this system].	1 2 3 4 5 6 7	Strongly Disagree	Strongly Agree

4.2. Analysis

Once data are collected, they need to be properly recoded, with a method that borrows from the SUS. Odd items are scored as [score - 1], and even items are scored as [7 - score]. As with the SUS, this removes the positive/negative keying of the items and allows a minimum score of zero. Each individual UMUX item has a range of 0 – 6 after recoding, giving the entire four-item scale a preliminary maximum of 24. To achieve parity with the 0–100 range provided by the SUS, a participant's UMUX score is the sum of the four items divided by 24, and then multiplied by 100. This calculation replaces the earlier methodology of weighting items by a 2.5 multiplier. These scores across participants are then averaged to find a mean UMUX score. It is this mean score and its confidence interval that become the application's UMUX metrics for a system's usability tracking and goal-setting.

4.3. Limitations

The UMUX, like the SUS, provides a subjective evaluation of a system's usability. Its scoring has yet to be compared to objective

metrics, such as error rates and task timings, in a full experiment. Additionally, as it is currently the first module in a planned series of user experience measures, it only measures usability.

As the UMUX consists of only four Likert items, it has fewer total data points available to respondents than the SUS, although the move to a seven-point scale does provide some mitigation. It has four seven-point items for a total of 28 data points, while the SUS has ten five-point items for a total of 50 data points. By comparison, a singular ease of use item like that used in Tedesco and Tullis (2006) may have only five data points. Reducing the total information capacity of a survey effort can result in a less sensitive measure. Once validity and reliability are established, there is still a potential risk of application beyond the metric's scope. For example, a simple ease of use item may do an exemplary job of measuring ease of use, but a particular user experience professional needs to determine whether that information is sufficient as a usability metric.

4.4. Conclusion

It can be concluded that the Usability Metric for User Experience is a reliable, valid, and sensitive alternative to the System Usability Scale. It correlates with the SUS at a rate higher than 0.80, its items align on one usability factor, and it is fully capable as a standalone subjective usability metric. It is also aligned to a fundamental learning for the user experience community: in order to measure user experience effectively, its components need to be measured efficiently. The compact size of the UMUX is suited to a more fully realized measurement model of user experience. Such a model would go beyond Product Use, and would include other product lifecycle stages such as Brand Awareness and Installation (Sward and MacArthur, 2007). Sward (personal communication, August 5, 2009) indicates significant progress in this area. The UMUX is well-positioned as a foundation for developing future instruments, with the ultimate goal of metrics that can target any of a product's user experience aspects in a way that is concise and cross-validated.

Acknowledgements

Thanks to David Sward of Symantec™ for his work on the user experience lifecycle model, Pete Lockhart of Intel® for item lan-

guage suggestions, Charles Lambdin of Intel® for statistical assistance, and Linda Wooding of Intel® for management support in implementing this research.

References

- Brooke, J., 1996. SUS: A "quick and dirty" usability scale. In: Jordan, P.W., Thomas, B., Weerdmeester, B.A., McClelland, A.L. (Eds.), *Usability Evaluation in Industry*. Taylor and Francis, London.
- Chin, J.P., Diehl, V.A., Norman, K., 1988. Development of an instrument measuring user satisfaction of the human-computer interface. In: *Proceedings of ACM CHI '88*, Washington, DC, pp. 213–218.
- Cox III, E.P., 1980. The optimal number of response alternatives for a scale: a review. *Journal of Marketing Research* 17, 407–422.
- Diefenbach, M.A., Weinstein, N.D., O'Reilly, J., 1993. Scales for assessing perceptions of health hazard susceptibility. *Health Education Research* 8, 181–192.
- Finstad, K., 2006. The system usability scale and non-native english speakers. *Journal of Usability Studies* 1 (4), 185–188.
- Finstad, K., in press. Response interpolation and scale sensitivity: evidence against five-point scales. *Journal of Usability Studies*.
- Hoyer, A., 1993. Methods for Evaluating user Interfaces. Cognitive Science Research Paper No. 301. School of Cognitive and Computing Sciences, University of Sussex.
- ISO 9241-11 (1998). *Ergonomic Requirements for Office Work with Visual Display Terminals (VDTs)*. Part 11: Guidance on Usability.
- Kirakowski, J., Porteous, M., Corbett, M., 1992. How to use the software usability measurement inventory: the users' view of software quality. In: *Proceedings European Conference on Software Quality*, Madrid.
- Lewis, J., 1995. IBM Computer usability satisfaction questionnaires: Psychometric evaluation and instructions for use. *International Journal of Human-Computer Interaction* 7 (1), 57–78.
- Lewis, J.R., Sauro, J., 2009. The factor structure of the System Usability Scale. In: *Proceedings of the Human-Computer Interaction International Conference (HCI 2009)*, San Diego CA, USA.
- Sauro, J., Dumas, J.S., 2009. Comparison of three one-question, post-task usability questionnaires. In: *Proceedings of the 27th International Conference on Human Factors in Computing Systems*. Boston.
- Sward, D., MacArthur, G., 2007. Making user experience design a business strategy. *Towards a UX Manifesto; SIGCHI Workshop*. Lancaster, UK, September 3–4.
- Tabachnik, B.G., Fidell, L.S., 1989. *Using Multivariate Statistics*, 2nd ed. Harper Collins, New York.
- Tedesco, D., Tullis, T., 2006. A comparison of methods for eliciting post-task subjective ratings in usability testing. *Usability Professionals Association (UPA) 2006*, 1–9.
- Tullis, T.S., Stetson, J.N., 2004. A comparison of questionnaires for assessing website usability. In: *Proceedings of UPA 2004*, June 7–11.

The UX Metrics Table: A Missing Artifact

Dieter Wallach^{1,2(✉)}, Jan Conrad¹, and Toni Steimle²

¹ University of Applied Sciences, Kaiserslautern, Germany

{dieter.wallach, jan.conrad}@hs-kl.de

² Ergosign Switzerland AG, Zurich, Switzerland

toni.steimle@ergosign.ch

Abstract. User Experience Design approaches typically rely on the creation of various concrete artifacts that are constructed and refined during the iterative course of an UX project. While the respective details and, correspondingly, the labels to designate resulting artifacts might vary, we often find Personas and Scenarios for consolidating insights in the Research Phase of a project; Scribbles, Wireframes or Mockups for spelling out our ideas in the Design Phase; and the use of (interactive) prototypes in an Evaluation phase, where we document usability findings in Usability Reports. In this paper, we introduce the concept of a UX Metrics Table, a comprehensive artifact that supports UX designers by guiding their project activities and by helping to derive an informed decision about the termination of iteration cycles. To exemplify the use of a UX metrics table and example is presented showing the application of the UX metrics table in a summative evaluation project.

Keywords: UX metric · UX design · User experience · Human-centered design · Iteration · Lean UX · System usability scale · Formative evaluation · Summative evaluation

1 Introduction

The term *User Experience (UX)* is, despite being devised almost 25 years ago by Donald Norman, still a somewhat chatoyant concept. In an approach to provide an overview of its meaning, the site allaboutux.org lists 27 definitions of user experience that were established until 2010. The ISO 9241-210 defines user experience as “a person’s perceptions and responses that result from the use or anticipated use of a product, system or service ... [UX] includes all the users’ emotions, beliefs, preferences, perceptions, physical and psychological responses, behaviors and accomplishments that occur before, during and after use ... Usability criteria can be used to assess aspects of user experience”. While this definition falls short of explicitly mentioning directly measurable UX metrics, the situation is more relieved in the ISO 9241-110 definition of usability. It defines usability as “the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use”. While the exact relation of usability to the obviously more comprehensive concept of user experience is not spelled out extensively in the ISO 9241 definition, we are informed that “usability criteria can be used to assess aspects of user experience”. Completion rates or error rates can be used to

operationalize the usability criterion of effectiveness, while time on task is an example for a quantification of the efficiency criterion. Completion rates and time on task can be measured objectively, while the third usability criterion, satisfaction, is a subjective measure that can be elicited on the task level using the *System Usability Scale*, for example (Sauro 2011). Operationalizing and including criteria in the measurement that extend the task level (on which the semantically tighter definition of usability mainly operates) as required by the ISO definition of user experience is clearly more complex: taking the subjective consequences of an “anticipated use of a product” or a “user’s emotions, beliefs, preferences, perceptions” into account demands the application of valid and reliable methods like the *User Experience Questionnaire (UEQ)* (Laugwitz et al. 2008).

In this paper we introduce an artifact called UX metrics table that focuses on the identification, measuring and comparative interpretation of UX metrics. To pave the ground for outlining a UX metrics table, some essential terms are clarified in the next section.

1.1 What Do We Mean by UX Metrics?

A metric is understood as an approach to the measurement or evaluation of a phenomenon under consideration (Tullis and Albert 2013). A metric that focuses on the usability of an interactive system like *time on task* can be measured directly and objectively by the assignment of a numerical value representing the delta between the start time and the end time of working on a task for a given sample of users. We need to determine an instrument for measuring, as well as the units for reporting the results of its application in a (set of) situation(s). A potential result would be a statement like: *The mean time for completing an order with system X is 37.9 s (with a standard deviation of 7.2 s)*.

When moving from the definition of usability to its superordinate concept of user experience according to the ISO 9241 we fall short of simple, directly observable indicators and need to resort to subjective measures like the UEQ mentioned above or to marketing metrics like brand perception (Sauro 2015) to capture specific aspects of user experience.

UX metrics, as results of measurements are *quantitative* by nature: we can compare the measurement results of a certain UX metric for different systems or contrast the measurement before and after the redesign of a system. In order to qualify as a well-founded and useful UX metric, the respective measure is required to be *valid* (i.e. it should measure what it claims to measure), *reliable* (i.e. it should produce similar results under similar conditions) and *objective* (i.e. it should be independent of the person conducting the measurement and should be free of references to outside influences). Ideally, a UX metric should be *easy* and *economically to measure* and its results should be *understandable* and *informative*. Embedding an instrument for measuring UX in a comprehensive framework of UX like the Components Model of User Experience (Thüring and Mahlke 2007) provides valuable theoretical underpinnings to support a sound interpretation of its results (see the meCUE questionnaire, Thüring and Minge 2014). To aid broad applicability, metrics should be *flexible* for

utilization in early design phases and should give meaningful results even with *small sample sizes*.

Leech (2011) argues that in order to be helpful, a UX metric needs to come with a *timescale* (to designate the temporal period under consideration), a *benchmark* (to allow for comparisons), a *reason to be reported* (to focus on significant data) and an *action* (that would allow appropriate response in light of available data) associated.

1.2 Why Should We Collect UX Metrics?

UX metrics provide directions for design. Depending on the requirements of a given project, the goal of design activities might vary: in one project we might primarily focus on achieving an efficient interaction concept while we might need to balance (conflicting) metrics like efficiency, error rates and/or learnability in the next project.

Identifying a relevant set of metrics and adequately balancing conflicting metrics is crucial to the success of product development. Starting with the vision of a new or improved product, we need to identify UX metrics to qualify what we mean by a desirable user experience in the respective project context. Explicit UX metrics help to shape and elaborate the goals of product development. Agreeing on benchmarks that come as target (and, optionally, acceptance) values for the UX metrics selected fosters insightful discussions especially in cross-functional teams and aligns project members on a strategic level. UX metrics guide data collection during the research phase of a project by pointing to empirical information required to come to informed design decisions.

Establishing UX metrics also helps us to evaluate our assumptions: they aid the interpretation of conducted studies, measure the impact of changes to a product and spell out improvements over iteration cycles. UX metrics tell us when to stop iterating because target values are met or exceeded. Defined target values for UX metrics allows a calculation of the intended benefits when conducting predictive Return-on-Investment (ROI) analyses. Monitoring the status and continuously reviewing the relevance of the chosen UX metrics acknowledges the dynamics of user experience and provides opportunities for prompt action in the light of changing UX requirements.

UX metrics make it easy to communicate project progress — and provide convincing numbers to prove it. Finally, UX metrics allow for easy comparisons of different products or different versions of the same product. Tullis and Albert (2013, p. 8), in their seminal book on UX metrics claim that “Metrics add structure to the design and evaluation process, give insight into the findings, and provide information to the decision makers”.

In defiance of the outlined value, UX metrics are often neither explicitly operationalized, nor agreed upon or annotated with target and/or acceptance values. Nielsens (2001) infamous quote “Metrics are expensive and are a poor use of typically scarce usability resources” might not be completely inculpable for that. A fortunate, clear exception are approaches that sail under the Lean UX flag following a designated *build, measure and learn loop* (Goethelf and Seiden 2012). In too many projects, however, UX metrics remain implicit and lose their guiding force. Even worse, team members, who are jointly contributing to the development of a product, might individually be striving

for the attainment of different (or even conflicting), but unexpressed UX goals. The lack of explicit and shared UX metrics impedes a smooth orchestration of project activities and misses opportunities to focus on informed, goal-oriented actions.

User experience design approaches typically rely on the creation of various concrete artifacts that are constructed and refined during the iterative course of an UX project. UX metrics need to be tightly engrained in the landscape of core UX artifacts to become widely understood as a matter of course and used as informative tools for design. A UX metrics table connects personas and scenarios with the explicit elaboration of quantitative metrics and has proven to be a helpful artifact through the stages of a UX project. In the next section we discuss ways to establish a UX metric.

1.3 Establishing UX Metrics

Meaningful UX metrics focus on the critical interaction scenarios of using an application. A scenario might be critical for different reasons: it is used very frequently, errors imply severe consequences, the touch point of interaction might contribute crucially to the general impression of the application, the interaction sequence might be of eminent importance for learning how to use a system or other essential scenarios that shape the user experience significantly. We first need to identify these critical interaction scenarios in order to set concrete expectations regarding relevant UX metrics.

Different scenarios may be critical for different *personas* that are conceptualized as representative users of an application. When we establish UX metrics we need to take critical scenarios and their associated personas into account. Independently from the concept of a UX metrics table presented in this paper, Travis (2011) suggested a related approach in which he suggests to first identify the *red routes* as an initial step to create UX metrics: “Most systems, even quite complex ones, usually have only a handful of critical tasks. This doesn’t mean that the system will support only those tasks: it simply means that these tasks are the essential ones for the business and for users. So it makes sense to use these red routes to track progress”. Travis refers to *user stories* (Cohn 2004) to support “thinking of the needs and goals of a specific persona” and to “fully ground” the scenario in context.

Including personas and scenarios in the formulation of a UX metric helps to prevent the definition of overly generic UX metrics that would require a significant operationalization before they can be associated with a method for measurement: “... «easy to use», «enjoyable» or «easy to learn»”. It’s not that these aren’t worthy design goals, it’s that they are simply too abstract to help you measure the user experience” (Travis 2011). Defining UX metrics in reference to personas and scenarios gives us the concreteness needed to declare successful goal attainment.

What is missing yet in order to arrive at *quantifiable* UX metrics are numerical values that form precise criteria for comparison. We may want a revised system to be *better* with regard to a certain UX metric than its predecessor, and/or we want it to be *as least as good* as its next competitor. We need to know *benchmark values* for comparison to indicate success or failure. In a summative evaluation, we can then qualify a measured value for a relevant UX metric to be *good* when it exceeds the benchmark value — or to be *bad* when it is lower, considering just a simple case.

Having benchmark values for UX metrics does not just allow for meaningful comparisons, but also helps to set reasonable *target values* regarding the magnitude of intended improvement. Arriving at relevant benchmark values can be achieved by measuring a UX metric with regard to a preceding system version, a competitor's offer — or, for some UX metrics, even with the values for the manual processing of some up-to-now unsupported task. Technical capabilities and insights from research activities typically inform the precise assignment of target and acceptance values. If in need of available benchmark values, generic average values for UX metrics, as published by Sauro (2012), can be used as rough first evidence.

In practice, differentiating between intended *targetvalues* and *acceptance values* has turned out to be helpful in some occasions. Target values represent true indicators of success, while achieving acceptance values points to the right direction but leaves room for improvement.

Identifying and prioritizing UX metrics is typically not a solitary endeavor. Different UX metrics might be of varying importance to different stakeholders, it is thus advisable to engage all interested parties when selecting and consolidating the set of UX metrics considered to be relevant. Especially in early phases of a project, large-scale UX goals, as spelled out in Google's HEART framework (Happiness, Engagement, Adoption, Retention, and Task Success, see Luenendonk 2015) are often put forward and later refined by referring to measurable metrics. Tullis and Albert (2013) extensively discuss behavioral and attitudinal UX metrics, categorized as performance metrics, issue-based metrics, self-reported metrics, as well as behavioral and physiological metrics.

2 The UX Metrics Table

Human-centered design activities are, by their very nature, artifact-centered. The construction of artifacts is the lowest common denominator uniting the iteratively intertwined phases of a human-centered product development cycle. Coming in different guises and named according to different flavors, artifacts are shared as communication tools amongst stakeholders, are evaluated and refined and, if necessary, abandoned. The maturity of artifacts indicates, within the limits of iterative approaches, progress in UX projects: We envision future users of an interactive system by establishing lively personas as representative archetypes. Scenarios excite the working goals of an acting persona in context and constitute an essential ingredient for deriving requirements. Scribbles, wireframes, user journeys and interactive prototypes let a product gradually come alive and support an early experience of a product's essentials. In contrast to the artifacts mentioned before, UX metrics have, however, not yet found a firm home in the artifact arsenal of UX professionals.

A *UX metrics table* links personas and scenarios to explicit UX metrics in a comprehensive artifact accompanying human-centered design activities. It continuously conveys the targets for the design, helping to keep the focus on agreed upon quantitative UX goals of a project. The rows of a UX metrics table refer to the UX metrics considered, presented in decreasing order of priority. Its columns provide the agreed upon parameters of the metric.

Figure 1 shows the header of an UX metrics table. The UX metrics table consists of nine columns that define (1) the *UX metric* under consideration, (2) the method to measure the UX metric, (3) the *persona* representing the intended user, (4) the *scenario* that provides the situational context, (5) a *benchmark* value for comparison, (6) a *target value* (complemented by an optional *acceptance value*) that serves as a quantitative goal for this metric, (7) a *result* column, holding the measured score for the UX metric that was empirically measured, (8) a *time scale* column to indicate the time frame within which the intended result is to be achieved and (9) a *sample* column to describe the designated sample for evaluating the UX metric.

UX metric	Measure	Persona	Scenario	Benchmark	Target	Result	Time scale	Sample
-----------	---------	---------	----------	-----------	--------	--------	------------	--------

Fig. 1. Elements of a UX metrics table

The number of rows representing the chosen UX metrics and their respective content depend of course on the specific goals and circumstances of an actual project. While starting with a small and focused list of UX metrics is advisable, the details of a UX metric table are subject to change over time. In practice, UX metrics might be added, rows where the *result* value equals or exceeds the *target* value might be highlighted to indicate success, and values might be adjusted in the light of new insights. The history of a UX metrics table represents dynamic snapshots of the progress made in establishing and attaining UX metrics.

Using the UX metrics table in real-world projects is quite straightforward: after having arrived at an initial understanding of the project goals, a first version of a UX metrics table is established in a joint meeting that brings relevant stakeholders together. Agreeing on relevant UX metrics paves the ground for a shared understanding of the project goals — that often get into a vivid tug-of-war for the “right” metrics and/or their prioritization. In projects targeted at the development of new products, initial UX metrics tables often start with tables consisting of hardly more than two concerted entries in the UX metric column that have mutually been agreed to be significant.

Setting UX metrics for a product requires (well-grounded) assumptions regarding the quality of its use. Establishing a UX metrics table gives additional weight to the careful creation of personas and scenarios based on empirical data. If user research has already provided evidence for their construction, the columns for *persona* and *scenario* can be filled — with their content certainly having a major impact on the definition of the UX metric under consideration. If not, the cells for persona and scenario in the UX table will remind the team about required action for further research to empirically ground the UX metric.

To illustrate the construction process of a UX metrics table, we discuss a simple example in the next paragraph.

3 Applying the UX Metrics Table in Practice

A UX Metrics Table provides a clear rationale for assessing the progress made during the iterations of a design project. It is easy to understand the benefits of the suggested artifact in the *build-measure-learn loops* of Lean UX approaches where (real world) validations of incrementally extended products are *formatively* conducted (see Steimle and Wallach, in preparation, for a discussion of UX metrics tables in Lean Development). Seeing how a UX metrics table can contribute to *summative* evaluations is less obvious. In the following real-world example we illustrate a summative use of UX tables in a project that was initiated by a client interested in an overall evaluation of a complex software system. For reasons of anonymization, we will denote the software as Wcs in the remainder of the paper.

3.1 An Example for the Summative Use of the UX Metrics Table

Wcs was installed three years ago with a large, international user base as the successor of an application with similar functionality. After having received negative user feedback regarding several UX deficiencies of Wcs, the client commissioned a summative usability study to inspect the application.

Analysis of the user base revealed two distinct user groups of Wcs:

- Expert users, who (1) mostly report to use Wcs daily or on several days per week, (2) qualify themselves to be very proficient in using Wcs (self-reports using a 7-point scale [1–7], yielding an average of 5.28 out of 7, SD = 1.63), (3) work on task sets that require full access to Wcs's functionality;
- Occasional users, with (1) >75% of this group using Wcs less than once per month, (2) qualify their level of proficiency comparatively low (self-reports using a 7-point scale [1–7], yielding an average of 3.66 out of 7, SD = 1.70), (3) work on restricted tasks with only limited access to Wcs's functionality.

Both user groups work with Wcs for about the same time (Expert users: 31.22 months, SD: 10.68; occasional users: 30.44 months, SD: 10.98).

To better understand the respective user attributes, their working goals and situational parameters, a Contextual Inquiry was conducted. The domain of Wcs is quite knowledge-intense, so the gathered insights were necessary prerequisites for a detailed UX inspection of the system. Results from the Contextual Inquiry were used to empirically ground the construction of two personas. The name *Emily Expert* was used to denote the Expert persona, while the group of occasional users was archetypically depicted by a persona called *Tim Sometime*.

When the client commissioned the UX evaluation, he was mainly interested in (a) an understanding of the user perspective, i.e. the subjective impression of Wcs's quality of use and (b) an expert opinion regarding reported (potential) usability flaws of Wcs.

Following the categorization according to Tullis and Albert (2013), the subjective usability impression of Wcs's users can be considered as a *self-reported metric* that can be measured, for example, using the *System Usability Scale (SUS)*. The SUS is a questionnaire to elicit the subjective impression of the usability of an interactive

system. It was published by Brooke (1996) and is used in an exceptionally large number of scientific and practical studies. TheSUS has excellent values for validity and reliability (Sauro 2011) and allows economical data elicitation since the questionnaire is comprised of only ten items. In response to these items, participants express their (dis)agreement with a system on a five-tier Likert-scale (example: “*I feel very confident in using Wcs*”, (*strongly disagree ... strongly agree*). A Susscore is calculated from the answers to the SUS questions and can range in 41 increments from 0 to 100. Zero symbolizes the theoretical lowest score and 100 the highest possible Susscore.

3.2 Determining the Values of the UX Metrics Table

Given the *UX metric* of a subjective usability impression, the Sus as a method for *measurement* and the two *personas*, we can already complete parts of Wcs UX metric table. The *scenario* cell in the table can be left empty since the Sus score refers to the overall impression of a system and is not related to an isolated scenario. To determine a *benchmark* value for the Sus score, data reported by Sauro (2011) was used. Sauro published Susscores from a total of 446 studies with more than 5,000 participants to derive general benchmark data that supports a comparative interpretation of obtained Susscores. The mean Sus score representing the entirety of those studies is 68, which can be inserted as the *benchmark* value for Wcs — with the *target* value set to >68. While this setting seems to be appropriate for the *Tim Sometime* persona representing occasional users, the values for *benchmark/target* is set in reference to Sauro’s mean Sus score for “*Internal-productivity software: Customer Service and Network Operations applications*”, which is 76.7 (SD = 8.8). In fact, all expert users of Wcs are working in the very same organization that released Wcs, while occasional users are employed in a variety of different organizations: the respective reference points are set accordingly. The slot for *time scale* was set to *Now* because the summative study is intended to capture the current state of the metric. Although the Sus is a robust tool even with small sample sizes, a web-based presentation of the Sus promised to gather data from a larger *sample size* (N > 50). Figure 2 shows the definition of the initial version of the UX metrics table for Wcs.

UX metric	Measure	Persona	Scenario	Benchmark	Target	Result	Time scale	Sample
Subj.Impr.	SUS	Emily E.	–	68	> 68		Now	> 50
Subj.Impr.	SUS	Tim S.	–	76	> 76		Now	> 50

Fig. 2. Initial UX metrics table for Wcs

Note: For reasons of simplicity and illustration, the table in Fig. 2 just comprises a single metric, referring to two different personas. Typically, a UX metrics table

includes various types of UX metrics that require carefully balancing (see Steimle and Wallach, in preparation).

3.3 Results

Data collection using a web-based version of the Sus was supported by using unique, computer-generated tokens to rule out multiple participation of the same person. Approximately 4,000 users of both user groups were invited via email to participate in the study.

A total sample of close to 500 Wcs users answered the Sus questionnaire, resulting in a data set of 414 occasional users and 75 expert users. The average Susscore after completing the Sus for occasional Wcs users was 40.4 (SD: 19.4). The confidence interval (95%) ranges between 38.48 and 42.22. Evaluating *Cronbach's alpha* to measure the internal consistency of the scale returned an excellent value of 0.914. The average Susscore for expert users was 43.3 (SD: 18.6). The confidence interval (95%) ranges between 39.02 and 47.58. Evaluating *Cronbach's alpha* to measure the internal consistency of a scale returns a very good value of 0.885.

The results for both user groups indicated in Fig. 3 indicate extraordinarily low Sus scores, expressing very negative subjective usability impressions about Wcs. The results were clearly confirmed by the outcome of another (ISO 9241-related) usability inventory (ISONORM, see Prümper and Anft 1997) that was jointly applied with the System Usability Scale in the sample. The highly significant positive correlation coefficients (*Pearson*, $p < 0.001$) of 0.78 (expert users) and 0.61 (occasional users) indicate a convergent validity of the two measurement tools. The picture is consistently completed by the results of a Heuristic Analysis targeted at identifying usability flaws with Wcs. A total of 48 findings was reported, with 6 findings classified to be of *minor* importance, 21 were categorized as *severe* and 21 as *critical*.

UX metric	Measure	Persona	Scenario	Benchmark	Target	Result	Time scale	Sample
Subj.Impr.	SUS	Emily E.	-	68	≥ 68	40.4	Now	75
Subj.Impr.	SUS	Tim S.	-	76	≥ 76	43.3	Now	414

Fig. 3. Complemented UX metrics table for Wcs

Although the study reported was summative, the negative evaluation results have convinced the client to start a project to redesign Wcs. With the basic UX metrics table shown in Fig. 2 as a quantitative vantage point, two metrics, *completion time* for the core scenario and *error rate* were added and the *time frame* set to the release date of the next version of Wcs. At present, log file analyses are carried out to determine *benchmark* values for the *completion time* and *error rates* regarding the selected core scenarios of Wcs.

4 Discussion

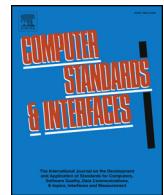
In this paper we have introduced an artifact coined UX metrics table that has proven to be very helpful in real world projects. Constructing a UX metrics table highlights the identification and tracking of UX metrics in the course of a project and supports a comprehensive understanding of the mutual dependencies between UX metrics. Linking UX metrics to concrete scenarios and personas provides the right resolution level for their meaningful definition and measurement. Introducing UX metrics table put metrics into true effect and supports institutionalizing their use in organizations.

As an outlasting project artifact, UX metrics tables connect the research-, design- and evaluation phase(s) of a project and provide clear quantitative means to determine project success — or failure. Travis (2011) argues: “That is the strength of metrics-based testing: it gives us the *what* of usability”. While we wholeheartedly agree, qualitative methods give us the *why* of usability and UX and help to make sense of available data. It is in combination with qualitative data when quantitative approaches permit significant insights. Design, however, should never degrade to any form of convulsive focusing on metrics and data — or as Bowman (2009) has put it in his now famous farewell letter to Google: “I won’t miss a design philosophy that lives or dies strictly by the sword of data”.

References

- Albert, W., Tullis, T.: Measuring the User Experience: Collecting, Analyzing, and Presenting Usability Metrics. Elsevier, Amsterdam (2013)
- Bowman, D.: Goodbye, Google (2009). [http://stopdesign.com/archive/2009/03/20/goodbye-google.html\(2/10/2017\)](http://stopdesign.com/archive/2009/03/20/goodbye-google.html(2/10/2017))
- Brooke, J.: SUS: A “quick and dirty” usability scale. In: Jordan, P.W., Thomas, B., Weerdmeester, B.A., McClelland, A.L. (eds.) Usability Evaluation in Industry. Taylor and Francis, London (1996)
- Cohn, M.: User Stories Applied: For Agile Software Development. Addison Wesley, Redwood City (2004)
- Gothelf, J., Seiden, J.: Lean UX: Applying Lean Principles to Improve User Experience. O’Reilly, Beijing (2012)
- Laugwitz, B., Held, T., Schrepp, M.: Construction and evaluation of a user experience questionnaire. In: Holzinger, A. (ed.) USAB 2008. LNCS, vol. 5298, pp. 63–76. Springer, Heidelberg (2008). doi:[10.1007/978-3-540-89350-9_6](https://doi.org/10.1007/978-3-540-89350-9_6)
- Leech, J.: A big list of UX KPIs and Metrics (2011). https://www.cxppartners.co.uk/ourthinking/ux_roi_what_to_measure_and_what_to_expect/. 2 October 2017
- Luenendonk, M.: Complete Guide to Google’s HEART Framework for Measuring the Quality of UX (2015). <https://www.cleverism.com/google-heart-framework-measuring-quality-ux-user-experience/>
- Nielsen, J.: Usability Metrics (2001). <https://www.nngroup.com/articles/usability-metrics/>. 2 October 2017
- Sauro, J.: 32 ways to measure the customer experience (2015). <https://measuringu.com/32-ways-cux/>. 2 October 2017

- Sauro, J.: 10 benchmarks for user experience metrics (2012). <http://measuringu.com/ux-benchmarks/>. 2 October 2017
- Steimle, T., Wallach, D.: Collaborative UX Design (in prep.)
- Sauro, J.: A Practical Guide to the System Usability Scale: Background, Benchmarks, & Best Practices. Measuring Usability LLC, Denver (2011)
- Thüring, M., Mahlke, S.: Usability, aesthetics, and emotions in human-technology interaction. *Int. J. Psychol.* **42**(4), 253–264 (2007)
- Thüring, M., Minge, M.: Nutzererleben messen – geht das überhaupt? In: Begleitforschung Mittelstand-Digital (Hrsg.) Wissenschaft trifft Praxis. Usability betrieblicher IT-Anwendungen, pp. 45–53 (2014)
- Travis, D.: How to mange design projects with user experience metrics. http://www.userfocus.co.uk/articles/how_to_manage_design_projects_with_ux_metrics.html. 2 October 2017



A methodology to develop usability/user experience heuristics

Daniela Quiñones^{a,*}, Cristian Rusu^a, Virginica Rusu^b

^a Pontificia Universidad Católica de Valparaíso, Valparaíso, Chile

^b Universidad de Playa Ancha, Valparaíso, Chile



ARTICLE INFO

Keywords:
Usability
User experience
Heuristic evaluation
Usability heuristics
Methodology

ABSTRACT

Technology, software systems and human-computer interaction paradigms are evolving. Traditional usability heuristics do not cover all aspects of user-system interactions. Many sets of heuristics have been proposed, with the aim of evaluating specific application domains and their specific usability-related features. In addition, several sets of heuristics are used to evaluate aspects other than usability that are related to the user experience (UX). However, most authors use an informal process to develop usability/UX heuristics; there is no clear protocol for heuristic validation. This can result in sets of usability/UX heuristics that are difficult to understand or use; moreover, the resulting sets of heuristics may not be effective or efficient evaluation tools. This article presents a formal methodology for developing usability/user experience heuristics. The methodology was applied in practice in several case studies; it was also validated through expert opinions.

1. Introduction

Usability is typically defined as the "capability of being used", in other words, the capability of an entity to be used [1]. Usability can be designed into the product and evaluated by usability inspections or usability tests. Usability is part of the User eXperience (UX). UX includes all the users' emotions, beliefs, preferences, perceptions, responses, behaviors and accomplishments that occur before, during and after using a product [2]. In addition to usability, there are several UX aspects that are important to evaluate in a software application.

Heuristic evaluation is a usability inspection method that is widely used to identify usability problems [3]. A set of evaluators judge the product interface to determine whether it satisfies usability principles [4]. This method uses "heuristic principles" or "usability heuristics" to evaluate usability.

Several authors have proposed many sets of heuristics for evaluating specific application domains and their specific features. This is because traditional heuristics do not evaluate specific features of specific applications. The many existing research works that are focused on designing new sets of heuristics indicates the interest and importance of creating heuristics for specific usability problems in specific application domains. In addition, several sets of heuristics are used to evaluate not only usability, but also other important aspects that are related to the UX, such as learnability, playability, and adaptability.

However, in most of the reviewed studies, there is no evidence that a formal process or methodology had been used to develop usability/UX heuristics [5]. This is because there are no theories or models that

are appropriate for establishing heuristics for specific domains or for evaluating usability heuristics in terms of their applicability to a specific type of application. Most existing heuristics have been developed based on the researchers' experience or using methods that are usually employed for other purposes but adapted to create heuristics [6].

It is possible to summarize the development process of usability/UX heuristics in two stages: (1) extraction of information and (2) transformation of extracted information into heuristics [7]. However, there is no consensus about the most effective process for developing heuristics for specific application domains [8].

Many proposed usability heuristics are established as an extension or adaptation of existing heuristics (such as Nielsen's heuristics). The studies do not document or explain in detail the process that is followed to create the new set of usability heuristics. Most studies do not specify whether the process is formal or informal, or if the authors used a methodology for developing heuristics.

Although there are methodologies that support the process of developing usability heuristics [9–14], there is currently no clear protocol for heuristic validation. The whole process of developing usability heuristics has yet to be formalized [15]. In conclusion, there is no formal process for formulating, specifying, validating and refining usability/UX heuristics. To facilitate the process of developing heuristics, we propose a formal methodology with several stages for establishing usability/UX heuristics.

This document is organized as follows: Section 2 explores the theoretical background; Section 3 explains the need for a formal methodology for developing usability/UX heuristics; Section 4 shows the process that we followed to create the methodology; Section 5 presents the formal

* Corresponding author.

E-mail addresses: danielacqo@gmail.com (D. Quiñones), cristian.rusu@pucv.cl (C. Rusu), virginica.rusu@upla.cl (V. Rusu).

methodology for developing usability/UX heuristics; **Section 6** explain briefly each stage of the methodology, including examples for the national park website domain; **Section 7** presents the methodology's validation and results; and **Section 8** presents the conclusions and future works.

2. Theoretical background

The concepts of usability, user experience, heuristic evaluations and evaluation methods are briefly presented below. In addition, related work is described.

2.1. Usability

As stated in the ISO 9241-11 standard, usability can be defined as follows [16]: “the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use”.

This standard explains how to identify the necessary information to be considered when specifying or evaluating usability in terms of user performance and satisfaction [17]. The standard was updated and is still under review [1].

2.2. Usability evaluation methods

As Fernández et al. highlight, “a method of usability evaluation is a procedure composed by a series of well-defined activities for data re-collection related to end user's interaction with a software product and/or how a specific feature of this software product contributes in achieving a certain degree of usability” [18]. Usability evaluations can be classified in two categories:

1. Usability inspections. These are revisions that are made by evaluators – usually experts – based on their own judgment, without the participation of users.
2. Usability tests. These are examinations that include real users, who evaluate a working software product.

Usability inspections are methods that are based on the participation of evaluators in inspecting interfaces [19], whereas usability tests are methods where a user or a group of users are asked to run a working system prototype and evaluate it with the objective of collecting information to improve the software product's usability [19].

2.3. Heuristic evaluation

A commonly used usability inspection method is heuristic evaluation. It identifies usability problems and “measures” the usability degree according to usability principles or usability heuristics. Heuristic evaluation was proposed by Nielsen and Molich [20]. Usability experts inspect a product (interface) based on heuristics to identify usability problems. The problems are classified (associated to the set of heuristics that is used) and rated (in terms of severity, frequency and critically).

2.4. Usability heuristics

Heuristic evaluation is performed based on a set of usability heuristics. They are called “heuristics” because they are more rules of thumb than specific usability guidelines [21]. Nielsen's 10 heuristics [21] are widely used to evaluate usability through heuristic evaluation. The set of Nielsen's heuristics is shown in **Table 1**.

Each system or application has features that differentiate it from other (types of) applications. For this reason, traditional heuristics may not assess unique features for specific domains, thereby neglecting important elements that must be evaluated. To effectively evaluate a specific domain, several authors have designed new sets of usability heuristics, by adapting Nielsen's heuristics and/or adding new heuristics to evaluate specific aspects.

Table 1
Nielsen's heuristics.

Id	Name
H1	Visibility of system status
H2	Match between system and the real world
H3	User control and freedom
H4	Consistency and standards
H5	Error prevention
H6	Recognition rather than recall
H7	Flexibility and efficiency of use
H8	Aesthetic and minimalist design
H9	Help users recognize, diagnose, and recover from errors
H10	Help and documentation

2.5. User eXperience

UX extends the usability concept beyond effectiveness, efficiency and satisfaction. According to the ISO 9241-210 standard, UX can be defined as follows: “person's perceptions and responses resulting from the use and/or anticipated use of a product, system or service” [2]. It states that UX “includes all the users' emotions, beliefs, preferences, perceptions, physical and psychological responses, behaviors and accomplishments that occur before, during and after use”.

In addition, the ISO 9241-210 standard remarks that UX “is a consequence of brand image, presentation, functionality, system performance, interactive behavior and assistive capabilities of the interactive system, the user's internal and physical state resulting from prior experiences, attitudes, skills and personality, and the context of use” [2].

2.6. User eXperience evaluation methods

UX evaluation methods are focused on determining how users feel about the targeted software system [22]. UX evaluation allows designers to understand and gain insight into how users perceive and value products. Having this understanding will help in achieving positive UX and desirable products [23].

Depending on what one wants to evaluate, UX evaluation methods can be classified into four categories [24]: (1) field studies, (2) laboratories studies, (3) online studies, and (4) questionnaires and scales. Since UX extends the usability concept, some UX aspects may be assessed through usability evaluation methods, such as heuristic evaluation according to heuristics that cover some of the UX dimensions. However, evaluating all UX aspects is much more challenging. Performing user tests is very critical and important in evaluating UX.

2.7. Related work

Rusu et al. [9] propose a methodology with six stages for developing usability heuristics. This methodology has been used by several authors to create new sets of heuristics for specific application domains [25–38]. These authors emphasize that the methodology facilitates heuristic design and specification since it recommends stages that support the whole development process. In addition, the methodology suggests a template for specifying the heuristics and includes a validation and refining stage. However, some stages do not clearly explain what activities should be performed (specifically, the descriptive and correlational stages); it is not explained how, when, and under what circumstances to iterate the stages.

Van Greunen et al. [10] suggest a three-phase process for creating heuristics for specific application domains. This methodology has been used in one study to develop new heuristics [39]. The authors present a formal process for developing heuristics. Each stage is explained in detail, and the activities to be performed in each one are clearly described. However, the methodology lacks a specific explanation of how to specify heuristics (it does not propose any template), the validation stage does not clearly explain how to use the tool that is proposed by

the authors, and previous experience in using heuristics is not considered when selecting evaluators for the validation stage.

Hub and Čapková in [11] proposed a methodology for creating a suitable set of heuristics for a specific application domain: a public administration portal. The methodology has not been used in other studies. The authors consider different roles for the creation of heuristics: (1) experts for specification and revision of heuristics and (2) end users for evaluation and revision of the heuristics. The authors present an interesting structure of work, since they include different participants in the process of heuristic development and revision. However, they do not present a formal validation of heuristics. There are no details on how to validate heuristics or prove that they are effective.

Franklin et al. in [12] propose a methodology for establishing heuristics for a particular application: collaborative augmented reality remote systems. The methodology was created for developing heuristics for that specific application and has not been used in other studies. The authors propose a methodology for adapting existing heuristics and creating a new set of heuristics, including activities such as review, selection, deletion, fusion and adaptation. For validation, they propose using a focus group of Human–Computer Interaction (HCI) specialists (experts) to review the heuristics. Based on the obtained recommendations, the heuristics are refined. However, it is not explained in detail how to specify new heuristics since the proposed methodology is based mainly on adapting existing heuristics.

Lechner et al. [13] present a methodology for developing a set of usability heuristics. The methodology proposes two phases with different activities. Until now, this methodology has not yet been used to create new heuristics. The authors suggest a validation process that involves only real users, not evaluators (experts). In the first phase, the experts work together to define the new set of heuristics that are based on existing heuristics. Then, in the second phase, the heuristics are validated with users (users rate heuristics based on their perceived applicability). The methodology does not include experts in the validation process. However, final users may not have the knowledge to determine whether a heuristic is correctly specified or not. In addition, the phases are not iterative and a clear procedure for specifying and refining the heuristics is not proposed.

Hermawati and Lawson [14] suggest a methodology about how to expand heuristics sets for a specific application domain and validate them. The methodology is divided into two stages and includes users in the process of developing heuristics. Nevertheless, some activities are unclear, that is, information is lacking on how to perform certain actions: (1) the stages do not clearly explain the activities to be performed and in what order to perform them; (2) there is no explanation on how to define or specify the new heuristics (a template is not provided); (3) the authors do not indicate how to refine the set of heuristics after validation; (4) the validation methods that should be applied are not clearly proposed; and (5) there is no recommendation on the type of users that should participate in the validation process.

The reviewed methodologies present interesting, novel and useful approaches for developing heuristics. However, they lack formal specifications of stages and activities and most do not present a clear and well-defined process for validating the heuristics. In conclusion, there is still no methodology that proposes a formal and systematic process for specifying and validating usability/UX heuristics.

3. Need for a formal methodology

Several authors have stated that it is necessary to create new heuristics for evaluating specific applications. For example, in [40], the authors conclude that “the currently existing usability heuristics are too general to be applicable for evaluating the usability of mobile map applications”. Thus, it is necessary to develop a new set that evaluates the specific features of that domain. In [41], the authors observed that there were relatively small and inadequate sets of heuristics that could be used to evaluate Learning Management Systems, since these sets do not take into account the specific features of such systems (such as interface and didactic effectiveness). In addition, while existing

heuristics may be used for evaluations of a specific application domain, these can be too generic and their applicability to the domain may be limited, which, in turn, may impair any evaluation that uses them [42].

As a result, specialized sets of heuristics are continually being developed and published [43]. This is mainly because the existing heuristics are too generic to evaluate specific aspects of a new application, system or device.

As Hermawati and Lawson indicate in [7], Nielsen's heuristics can be used to evaluate user interfaces for various domains, but the heuristics must be adjusted to ensure that specific usability issues of certain domains' user interfaces are not overlooked. Nielsen's heuristics allow a significant number of usability problems to be identified, but they do not address the specific aspects that a particular kind of software application can have [44]. For example, after performing an experimental evaluation in [45], the evaluators indicated that Nielsen's heuristics do not cover all usability aspects of a transactional web application.

When creating new heuristics, the authors follow different approaches. However, most studies do not document or explain (in detail) how the approach has been applied or the process that they followed to develop new heuristics [5]. Following a formal process is crucial for developing an effective and efficient set of heuristics.

Although there are methodologies that support the process of developing heuristics [9–14], there is currently no clear protocol for heuristic validation. The whole process of developing usability heuristics has yet to be formalized [15]. Based on the above, we propose a formal methodology for developing usability/UX heuristics and clearly identifying all stages to be followed to formulate, specify, validate and refine a new set of heuristics.

We refer to usability and UX heuristics, as it is possible to establish heuristics whose purpose is to evaluate relevant UX aspects other than usability, such as playability, learnability, and accessibility. We propose a methodology that allows the creation of different sets of heuristics, based on the aspects, factors or attributes to be evaluated.

4. Process of creating the methodology

We used several inputs to create the methodology and we performed two iterations to formalize, validate and refine it. The inputs that were used and the iterations that were performed are explained below.

4.1. Inputs for creating the methodology

Fig. 1 shows the 3 inputs that we used to create and formalize the methodology, which is the output of the whole process.

The methodology is based on results obtained in the following steps:

1. Systematic literature review: We conducted a systematic review of the processes that are followed in different studies to develop and specify usability heuristics, along with the methodologies that have been proposed for developing usability heuristics [5].
2. Experimental results: We performed several interviews and surveys to determine the perception of evaluators who were using Nielsen's heuristics in a heuristic evaluation [46]. In addition, the methodology of

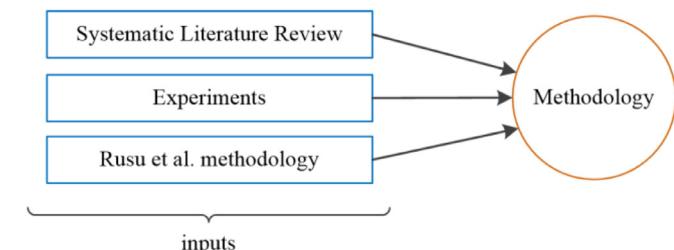
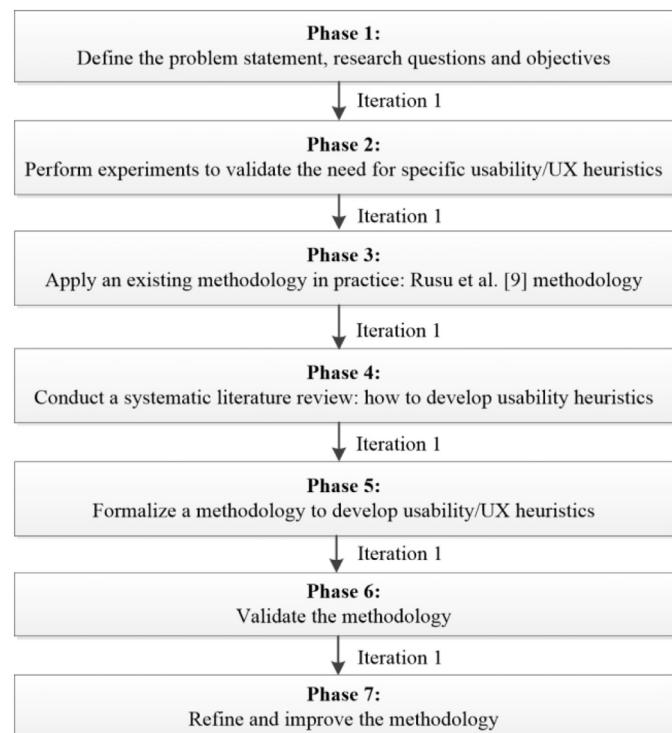


Fig. 1. Inputs for creating the methodology for developing usability/UX heuristics.

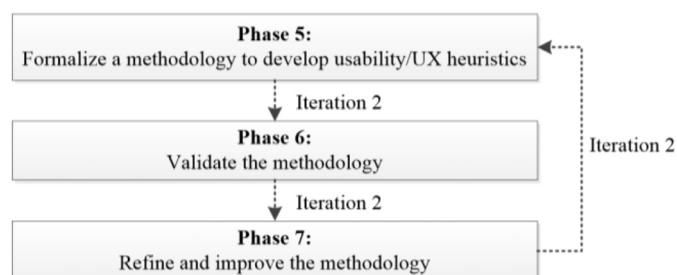
Table 2

Comparison between methodology that was proposed by Rusu et al. [9] and our methodology.

Step	Methodology of Rusu et al. [9]	Our methodology
Step 1: Exploratory stage	Collect bibliography for the main topics of the research: specific applications, their characteristics, and general and/or related (if identified) usability heuristics.	We improved the step definition. We included new types of information that must be collected, and we proposed information sources that can be reviewed.
Step 2: Experimental stage	–	New step. This step analyzes data that are obtained in experiments that were performed by other researchers that may provide additional information.
Step 3: Descriptive stage	Highlight the most important characteristics of the previously collected information, to formalize the main concepts that are associated with the research.	We improved the step definition. We explained in detail how to group the information that was collected in the previous stages and we proposed a three-level scale for prioritizing the grouped information.
Step 4: Correlational stage	Identify the characteristics that the usability heuristics for specific applications should have, based on traditional heuristics and case study analysis.	We refined and improved the step definition. We proposed to match application features, usability/UX attributes and existing heuristics to cover all these aspects with the new set of heuristics.
Step 5: Selection stage	–	New step. This step identifies the existing heuristics that can be used and/or adapted for the new set of heuristics, determines which existing heuristics can be discarded, and what heuristics should be created.
Step 6: Specification stage	Formally specify the set of proposed heuristics using a standard template. Template elements: ID, Name, Definition, Explanation, Examples, Benefits and Problems.	We refined and improved the step definition. We changed the name from "Explicative stage" to "Specification stage" (since the second term is clearer) and we added 5 new elements to the template.
Step 7: Validation stage	Check the new heuristics against traditional heuristics by experiments, through heuristic evaluations that are performed on selected case studies and complemented by user tests. The authors propose evaluating the set of heuristics in terms of the number of usability problems that are identified.	We refined and improved the step definition. We proposed an additional method for validating the new set of heuristics: expert judgment. Moreover, we added new elements to validate the effectiveness of the new set of heuristics through heuristic evaluations (5 criteria).
Step 8: Refinement stage	Refine the set of proposed heuristics based on the feedback from the validation stage.	We improved the step definition. We explained how to document the feedback that was obtained in step 7 (what heuristics to create, refine and/or delete, and why and how to do it, and what steps to repeat).

**Fig. 2.** Process of developing the methodology (first iteration).

Rusu et al. [9] was applied to create heuristics for transactional websites to detect deficiencies in the process of developing heuristics [47]. 3. Methodology of Rusu et al.: The methodology is based on [9]. We applied the methodology of Rusu et al. in practice [47] and analyzed the process that was followed by the authors to create heuristics for grid computing applications [25]. We made several changes with the aim of improving the methodology that was proposed by Rusu et al. [9]. Table 2 shows a comparison between the methodology that was proposed by Rusu et al. [9] and our methodology, and details what has been added and/or improved.

**Fig. 3.** Process for developing the methodology (second iteration).

The methodology was formalized at different levels through descriptions, tables and diagrams. We added new elements into the methodology specification: (1) we added BPMN diagrams to explain steps and activities (Appendices A–K) and (2) we included summary tables for each step to explain inputs, outputs and activities (Section 5).

4.2. Iterations

We performed two iterations to formalize, validate and refine the methodology. The process that we followed to create the methodology was divided into 7 phases.

4.2.1. First iteration

In the first iteration, all 7 phases were performed (see Fig. 2).

In phase 1, we defined the problem statement, research questions and objectives.

In phase 2, we performed several experiments to validate the need for specific usability/UX heuristics. We applied surveys and conducted interviews to determine if Nielsen's heuristics [21] are easy to understand for evaluators and allow the evaluation of all the (domain-related) features of a specific application. Experimental data show that Nielsen's heuristics present several problems in their definition [46] and that it is not possible to evaluate all the features of a specific application.

In phase 3, we applied the methodology that was proposed by Rusu et al. [9] to develop a new set of usability heuristics for transactional website applications [47]. We analyzed the methodology and its stages

Table 3
Step 1: Exploratory stage.

Step 1: Exploratory stage: perform a literature review.

What do I need to get started?	→ A specific application domain that needs a new set of heuristics or checklist.
What to do?	Collect information about the specific application domain, their features, the usability/UX attributes that will be evaluated with the new set of heuristics, and the existing set of heuristics (and/or other relevant elements, such as principles, guidelines, and patterns).
What is obtained?	<ul style="list-style-type: none"> ► ① Information about application (definitions and features). ► ② Usability and UX attributes. ► ③ Sets of heuristics and/or other relevant elements.

to identify potential deficiencies and suggest improvements.

In phase 4, we conducted a systematic literature review regarding the processes that are followed to develop and specify usability heuristics [5]. We analyzed different methodologies that have been proposed for creating heuristics and we concluded that they lack clear specifications of stages and activities and, especially, a clear procedure for heuristic validation [5].

In phase 5, we formalized a methodology for developing usability/UX heuristics. We made several changes with the aim of improving the methodology that was proposed by Rusu et al. [9] (see Table 2).

In phase 6, we performed a preliminary validation of an earlier version of the methodology, which is briefly described in [48]. The validation consisted of asking experts about what elements they would add, modify or remove to improve the methodology (see Section 6.1).

In phase 7, we refined the methodology based on the results and feedback that were obtained in the previous phase. All comments and recommendations were taken into account to improve the methodology (see Section 6.1).

4.2.2. Second iteration

In the second iteration, we performed a new validation and refinement of the methodology; phases 5, 6 and 7 were repeated (see Fig. 3).

In phase 5, we applied all the changes that were presented in phase 7 of the first iteration.

In phase 6, we conducted a survey of 9 usability/UX experts who applied the methodology to develop heuristics for several specific application domains. They were asked to evaluate each step of the methodology separately in 4 dimensions. We also included 5 questions that rate the methodology globally and 4 open questions that prompt qualitative comments (see Section 6.2).

In phase 7, we refined the methodology based on the expert opinions that were collected in phase 6. We analyzed all suggestions and we made changes accordingly (see Section 6.2). Section 5 presents the final version of the methodology.

5. Formal specification of the methodology to develop usability/UX heuristics

We created a methodology for developing usability/UX heuristics that includes 8 stages (see Fig. 4). The Business Process Model and Notation (BPMN) has been used to model each of the methodology's stages. BPMN provides a graphical notation for representing a business process as a diagram [49]. Appendices A–K show the BPMN diagrams for each stage of the methodology.

Although Fig. 4 shows sequentially the stages of the methodology, the development of heuristics may be performed iteratively. In specific situations (1), some stages may be optional, i.e., they are not performed for some reason; (2) some stages overlap, because it is necessary to perform two stages together; and (3) a stage may stop and one can go back to an earlier stage.

We recommend applying the methodology iteratively. The iterations allow the improvement of the set of heuristics and the

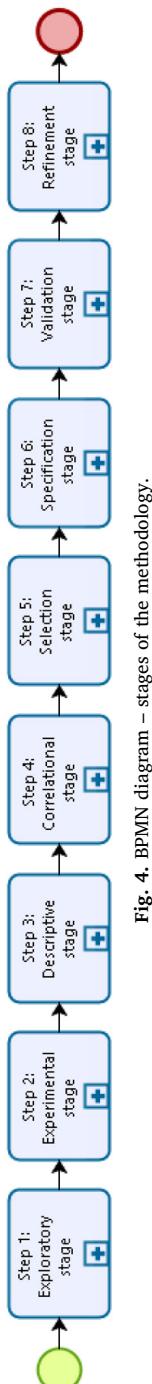


Fig. 4. BPMN diagram – stages of the methodology.

Table 4

Step 2: Experimental stage.

Step 2: Experimental stage: analyze data that are obtained in different experiments to collect additional information that has not been identified in the previous stage.	
What do I need to get started?	<ul style="list-style-type: none"> ↳ A specific application domain that needs a new set of heuristics or checklist. ■ The data can be obtained in previous experiments that were performed by other researchers. The obtained results will be related to specific features of the application, usability problems (both useful for Step 3: Descriptive stage) and problems with existing heuristics (useful for Step 6: Specification stage).
What to do?	<ul style="list-style-type: none"> ■ If there are no previous data, it is possible to perform experiments to obtain them, as long as there is time and evaluators are available to participate. In the case that it is decided to perform an experiment, the experiment might consist of Heuristic Evaluation, Usability Test, Interview and/or Survey. Consideration: This step is optional. The researcher can decide whether to analyze the data of previous experiments or conduct new experiments to collect useful and complementary information.
What is obtained?	<ul style="list-style-type: none"> ► ④ Additional specific features of the application ► ⑤ Detected usability problems ► ⑥ Problems with existing heuristics

Table 5

Step 3: Descriptive stage.

Step 3: Descriptive stage: select and prioritize the most important topics of all information that was collected in the previous stages.	
What do I need to get started?	<ul style="list-style-type: none"> ↳ ① Information about the application (definitions and features) ↳ ② Usability and UX attributes ↳ ③ Sets of heuristics and/or other relevant elements ↳ ④ Additional specific features of the application (optional*) ↳ ⑤ Detected usability problems (optional*)
What to do?	<ul style="list-style-type: none"> * The necessity of this step depends on whether experiments were performed in Step 2 1. Separate and group information according to the following topics: <ul style="list-style-type: none"> 1.1. Information about the specific application domain: definitions, classifications, context, areas of use and research justification. 1.2. Features of the specific application domain: elements that define the type of application (what the software product does and how it does it). 1.3. Usability and User Experience attributes: attributes that will be evaluated with heuristics. 1.4. Existing sets of heuristics and/or other relevant elements: traditional or specific sets of heuristics that were developed for similar applications and/or other elements that are relevant to the application (guidelines, patterns, and so on). 1.5. Usability problems detected (optional): problems identified by previous experiments. 2. Sort and prioritize the information of each topic according to the following scale: (3) highly important, (2) somewhat important, and (1) not important. <p>If after reviewing the existing heuristics, it is determined that the identified set properly evaluates the specific application domain, the researcher can stop the process of developing heuristics and use this set of heuristics.</p>
What is obtained?	<ul style="list-style-type: none"> ► ⑦ Selected information about the application ► ⑧ Selected features of the specific application domain ► ⑨ Selected usability/UX attributes ► ⑩ Selected sets of heuristics and/or other relevant elements

Table 6

Step 4: Correlational stage.

Step 4: Correlational stage: match the features of the specific application domain with the usability/UX attributes and existing heuristics (and/or other relevant elements).	
What do I need to get started?	<ul style="list-style-type: none"> ↳ ⑦ Selected information about the application ↳ ⑧ Selected features of the specific application domain ↳ ⑨ Selected usability/UX attributes ↳ ⑩ Selected sets of heuristics and/or other relevant elements
What to do?	<ul style="list-style-type: none"> ■ Find a match among features, usability/UX attributes and existing heuristics (and/or other relevant elements). ■ The matching process should be guided by features. Attributes should be associated with each feature. Each feature should have at least one matching attribute. Then, each feature and attribute is matched with the existing sets of heuristics. ■ Determine whether heuristics will be classified into categories. Each category can group heuristics that evaluate certain specific aspects. It is recommended to create categories as it helps reduce the complexity of the information that is collected.
What is obtained?	<ul style="list-style-type: none"> ► ⑪ Matched features, attributes and existing heuristics (and/or other relevant elements) ► ⑫ Categories

performance of new experiments to validate it. It is possible to repeat all stages, some stages, or only one stage of the methodology. The researcher may perform as many iterations as needed; the process is not always finalized with the Step 8: Refinement stage.

Each stage of the methodology is explained below. We present a summary table for each step, which shows the following: (1) definition of the stage; (2) the inputs that are needed to start the stage; (3) the activities that are performed in the stage; and (4) the outputs that are obtained at the end of the stage. The methodology is more extensive; however, to improve readability, a summarized version is presented (Tables 3–10).

6. Explaining the methodology

In this section, we briefly explain each stage of the methodology including examples for the national park websites domain.

6.1. Step 1: Exploratory stage

A review of the literature must be conducted to collect relevant information for developing the new set of heuristics. The researcher can review different information sources, such as scientific articles, theses,

Table 7

Step 5: Selection stage.

Step 5: Selection stage: keep, adapt and/or discard the existing sets of usability/UX heuristics that were selected in Step 3 (and/or other relevant elements).	
What do I need to get started?	↳ ⑩ Selected sets of heuristics and/or other relevant elements ↳ ⑪ Matched features, attributes and existing heuristics (and/or other relevant elements)
What to do?	For each heuristic, select one of the following options: 1. Keep the existing heuristic without any change: the heuristic is clear and correctly evaluates an aspect of the application and a usability/UX attribute. 2. Eliminate the existing heuristic: the heuristic evaluates aspects that are unrelated to the specific application. 3. Adapt the existing heuristic: the heuristic evaluates an aspect of the specific application and a usability/UX attribute, but changes are needed. It is also possible to combine two or more heuristics from different sets into a single heuristic. 4. Create a new heuristic: a new heuristic is required for evaluating a specific feature of the application (here, the heuristic is not created; it is only necessary to define the aspect for evaluation). Other relevant elements that were collected selected in Step 3 may be used to create a heuristic. Consideration: Eliminate redundant heuristics. Different sets may present similar or identical heuristics. ► ⑫ Classified heuristics (to keep, adapt, create and eliminate)
What is obtained?	

Table 8

Step 6: Specification stage.

Step 6: Specification stage: formally specify the new set of usability/UX heuristics.	
What do I need to get started?	↳ ⑬ Problems with existing heuristics (optional)* ↳ ⑭ Matched features, attributes and existing heuristics (and/or other relevant elements) ↳ ⑮ Categories (optional)** ↳ ⑯ Classified heuristics (to keep, adapt, create and eliminate)
What to do?	1. Define the number of heuristics. 2. Group heuristics into categories (if deemed necessary). 3. Determine which elements to include in the heuristics' specifications (such as id, definition, examples, checklists, etc.). 4. Formally specify heuristics. We propose using the following template: ■ Id: Heuristic's identifier. ■ Priority: Value that identifies how important the heuristic is in the evaluation of a specific aspect or feature. The value can be (3) Critical: Heuristic evaluates a crucial aspect; (2) Important: Heuristic evaluates a relevant aspect; or (1) Useful: Heuristic further improves the usability/UX. ■ Name: Heuristic's name. ■ Definition: A brief but concise definition of the heuristic. ■ Explanation: Detailed explanation of the heuristic. ■ Application feature: Feature or aspect of the specific application domain that is evaluated with the heuristic. ■ Examples: Examples of violation of and compliance with the heuristic. Include an image that graphically explains the problem. ■ Benefits: Expected usability/UX benefits when the heuristic is satisfied. ■ Problems: Anticipated problems of heuristic misunderstanding. ■ Checklist: Items or criteria that are associated with the heuristic. ■ Usability/UX attribute: Usability/UX attribute that is evaluated with the heuristic. ■ Heuristics related: Set (or sets) of heuristics on which the heuristic is based, along with the authors and the references. Considerations: ■ We recommend that the average number of heuristics be between 10 and 16. ■ If it has been decided to group heuristics into categories, they must be specified in a coherent order. ■ If it is necessary to add a greater level of detail, it is suggested to develop a checklist. ► ⑰ Set of proposed heuristics
What is obtained?	

Table 9

Step 7: Validation stage.

Step 7: Validation stage: validate the set of heuristics through several experiments in terms of their effectiveness and efficiency in evaluating the specific application.	
What do I need to get started?	↳ ⑱ Set of proposed heuristics Validate the set of heuristics through the following methods: 1. Heuristic evaluation: check the new set of heuristics against control heuristics (traditional or specialized heuristics) through heuristic evaluations. 2. Expert Judgment: check the validity of the proposed set of heuristics by asking to usability experts about their appropriateness for evaluating the specific application. 3. User Test: check whether the usability/UX problems that were identified in a heuristic evaluation using the new set of heuristics are usability/UX problems for users, and/or obtain the perceptions of users about a specific topic. Considerations: ■ We recommend always validating the set of heuristics through heuristic evaluation. ■ Expert judgment should be performed whenever possible, to receive additional feedback. ■ Complement the validation with user tests if necessary. ■ We recommend using more than one method to validate, to compare the results that are obtained by different methods.
What to do?	► ⑲ Heuristic evaluation results: effectiveness of heuristics ► ⑳ Expert judgment results (survey): utility, clarity, ease of use, need for checklist and comments about each heuristic ► ㉑ User tests results: users' perceptions
What is obtained?	

previous experiments (e.g., the results that were obtained in a heuristic evaluation), books, and websites (with reliable and credible information). [Appendix A](#) shows the BPMN diagram for Step 1: Exploratory stage.

6.2. Step 2: Experimental stage

This stage does not require additional explanations. [Appendix B](#) shows the BPMN diagram for Step 2: Experimental stage.

6.3. Step 3: Descriptive stage

The step aims at selecting and highlighting the most important topics of all information that was collected in the previous stage, to formalize the main concepts that are associated with the research. [Appendix C](#) shows the BPMN diagram for Step 3: Descriptive stage. [Table 11](#) shows the information that was collected and selected for creating usability/UX heuristics for national park websites.

6.4. Step 4: Correlational stage

In the matching process, it is possible that there is no existing heuristic that evaluates a specific feature of the application domain. In this case, the researcher only matches the feature with the usability/UX attribute. In Step 5 (Selection stage), a new heuristic that covers that feature and attribute will be tentatively proposed. [Appendix D](#) shows the BPMN diagram for Step 4: Correlational stage.

As an example, [Table 12](#) shows the matches among national park website features, usability/UX attributes, and the set of heuristics for virtual museums [52] that were selected in Step 3 (Descriptive stage). Some existing heuristics partially cover the features and attributes, while other features and attributes are not covered by any existing heuristics (this means that it is necessary to create new heuristics for evaluating these elements).

6.5. Step 5: Selection stage

[Table 12](#), which was created in Step 4: Correlational stage, can support the heuristic selection process and be used to determine whether to keep, adapt or eliminate each heuristic. Those heuristics that were matched with a specific feature and usability/UX attribute may be taken into account in the development of the new set of heuristics. [Appendix E](#) shows the BPMN diagram for Step 5: Selection stage.

[Table 13](#) shows some examples of the heuristic selection process for creating a new set of heuristics for national park websites. The entries in column “Applicability” indicate how important it is to consider the heuristic in the design of the new set. The applicability of the heuristic can have one of three values: (3) Critical, (2) Important, and (1) Useful (if included, much better). If a heuristic is eliminated, the last two columns of the table (“Aspect or feature covered” and “Applicability”) are left blank.

6.6. Step 6: Specification stage

The step requires the formal specification of the new set of usability/UX heuristics. We recommend keeping the number of heuristics relatively small (between 10 and 16). In the studies that were analyzed in the systematic review that develop new sets of heuristics [5], the authors create less than 20 heuristics. This is because it difficult to apply a large number of heuristics in practice. [Appendix F](#) shows the BPMN diagram for Step 6: Specification stage.

If it is necessary to add a greater level of detail, we suggest developing an additional checklist. A heuristic is a general rule. A heuristic evaluates a feature or specific aspect of an application. A checklist is more specific: it is a list of items that must be checked.

[Table 14](#) shows a set of heuristics for national park websites that was developed using the methodology. [Table 15](#) shows the complete template of heuristic NPH2: Multimedia resources.

6.7. Step 7: Validation stage

The step aims at validating the set of usability/UX heuristics. As it is a critical stage in the process, we detail the three types of validation that we propose. Experiments may be conducted to evaluate the effectiveness and efficiency of the new set of heuristics when evaluating the usability/UX of specific applications. [Appendices G–J](#) show the BPMN diagrams for Step 7: Validation stage.

6.7.1. Validation through heuristic evaluation

Perform a heuristic evaluation to evaluate the proposed set of heuristics against a set of control heuristics in specific case studies. For this, the researcher must carry out the following steps:

1. Select applications to evaluate: We recommend selecting a representative product for the specific application domain. For instance, for national park websites, a representative product should be the Yellowstone National Park Website (<https://www.nps.gov/yell/index.htm>).
2. Select the set of control heuristics: Control heuristics are a set of heuristics that will serve as the basis for comparing the results that were obtained in the heuristic evaluations. The researcher decides what set of control heuristics will be used for evaluation, according to the sets of heuristics that were identified in Step 1: Exploratory stage. If specific and/or related heuristics were not identified in Step 1, traditional Nielsen's heuristics [21] can be used as control heuristics. According to Anganes et al. [43], five attributes must be present in a set of specialized heuristics to determine their confidence: (1) Basis, (2) Validation, (3) Ease of learning, (4) Ease of use, and (5) Fit.
3. Select evaluators for heuristic evaluation: The selected specific application is evaluated by two separate groups of evaluators of similar experience under the same conditions (both groups evaluate the same product). One group uses only the set of heuristics that were defined in Step 6: Specification stage (called the Experimental group), while the second group uses only the set of control heuristics (called the Control group). The groups work separately. Then, the usability/UX problems that are identified by the two groups are compared. Evaluators should have similar experience conducting heuristic evaluations to obtain reliable results and eliminate external factors that may contaminate the experiment. Different levels of experience generate different results.
4. Evaluate the effectiveness of the heuristics based on the results that were obtained in the heuristic evaluations. To evaluate the effectiveness of heuristics, we recommend comparing the problems that were identified by both groups in terms of 5 criteria:
5. Numbers of correct and incorrect associations of problems to heuristics.
6. Number of usability/UX problems that were identified.
7. Number of specific usability/UX problems that were identified.
8. Number of identified usability/UX problems that qualify as more severe.
9. Number of identified usability/UX problems that qualify as more critical.
10. The new set of usability/UX heuristics performs well and it is an effective instrument when better results than the control heuristics are obtained in terms of (1), (2), (3), (4) and (5).

6.7.2. Validation through expert judgment

“Expert Judgment” is a term that refers to a technique in which the judgment of a person is based on their knowledge and experience in a specific area. We suggest asking experts about the usefulness, efficiency and effectiveness of heuristics to evaluate the usability of a specific application domain. In this case, the experts are divided in two groups: Researchers and Practitioners.

The Researchers are experts in HCI, usability and/or UX; they have experience designing or validating new sets of heuristics. The

Table 10

Step 8: Refinement stage.

Step 8: Refinement stage: refine and improve the new set of heuristics based on the feedback that was obtained in Step 7.		
What do I need to get started?	↳ ⑤ Heuristic evaluation results: effectiveness of heuristics ↳ ⑥ Expert judgment results (survey): utility, clarity, ease of use, need for checklist and comments about each heuristic ↳ ⑦ User tests results: users' perceptions (optional)*	* The necessity of this step depends on whether this validation was performed in Step 7.
What to do?	1. Document the problems that arose when using the set of proposed heuristics and the changes that should be made. 2. Define the heuristics to be created, refined and/or deleted, why, and how to do it. 3. Iterate and apply some stages again, if necessary.	
What is obtained?	► ⑧ Refining document: 1. What heuristics to create, refine and/or delete, why, and how to do it 2. What steps to repeat	

Table 11

Information that was collected and selected for creating usability/UX heuristics for national park websites.

Topic	Collected information	Selected information
1. Information about national park websites	Definitions of national parks (physical place) that were proposed by 4 different authors, their purpose and research justification (why it is necessary to create a new set of heuristics for national park websites).	There is no formal definition of national park websites, only definitions of national parks as physical locations. Thus, based on the literature review, a definition has been proposed for applications of this type. All general features were selected.
2. Features of the national park websites (general and specific features)	■ General features (crosswise features for any kind of software product that are also related to national park websites): 1) Feedback, 2) Visibility of contents, 3) Ease of navigation, 4) Error prevention, 5) Quick access, 6) Minimize the user's memory load, 7) Help for the user, and 8) Standards. ■ Specific features (features of national park websites, which differentiate them from other types of software products): 1) Updated information, 2) Virtual experience, 3) Multimedia resources, 4) Permissions, restrictions and recommendations, 5) Information credibility, 6) Asynchronous interaction, 7) Useful and interesting content, 8) Multi-language content, and 9) Connectivity with social networks.	All specific features were selected except <i>Connectivity with social networks</i> , since it is not related to national park websites.
3. Usability and UX attributes	■ Usability attributes that were proposed by the ISO standard [16]: effectiveness, efficiency and satisfaction. ■ Usability attributes that were proposed by Nielsen [50]: learnability, efficiency, memorability, errors and satisfaction.	The 5 usability attributes that were proposed by Nielsen were selected since they are more complete than those that were proposed by the ISO standard. Six factors that were proposed by Morville were selected (useful, usable, desirable, findable, credible and valuable). <i>Accessible</i> was not considered, as the evaluation accessibility is not intended with the new heuristics.
4. Existing sets of heuristics	■ UX factors that were proposed by Morville [51]: useful, usable, desirable, findable, credible, accessible and valuable. ■ Nielsen's 10 heuristics [21]. ■ Heuristics for virtual museums (15 heuristics) [52]	The set of heuristics for virtual museums [52] was selected. Nielsen's heuristics [21] were discarded because they are too generic and do not cover some specific features of national park websites. Virtual museum heuristics were developed based on Nielsen's heuristics, by adapting the generic aspects to the context of the application, which is similar to national park websites.

Table 12

Matches among national park website features, usability/UX attributes and existing heuristics.

Feature	Attribute	Heuristic
Updated information	- Satisfaction (usability)	–
- Useful, credible, valuable (UX)		
Virtual experience	- Satisfaction (usability) - Desirable, valuable, findable (UX)	VMH2: Visualization (partially covered)
Multimedia resources	- Satisfaction (usability)	VMH6: Resources and connections (partially covered)
Permissions, restrictions and recommendations	- Desirable, valuable, findable (UX)	–
- Useful, usable, findable (UX)	- Satisfaction (usability)	
Credibility in the information	- Satisfaction (usability)	VMH1: Visibility of system status (partially covered)
Asynchronous interaction	- Useful, credible, valuable (UX) - Satisfaction (usability)	VMH14: consistency and standards (partially covered) VMH15: Synchronous and/or asynchronous interaction (partially covered)
Useful and interesting content	- Useful, valuable (UX) - Satisfaction (usability)	–
Multi-language content	- Useful, usable, findable (UX) - Learnability (usability) - Useful, usable, credible (UX)	VMH4: Match between the system and the real world (partially covered)

Table 13
Heuristic selection process.

ID	Heuristic name/explanation	Action	Set of existing heuristics	Aspect or feature covered	Applicability
VMH1	Visibility of system status	Adapt	[52]	Feedback of processes or changes (general feature)	(1) Useful
VMH2	Visualization	Adapt	[52]	Adequate visibility of contents, virtual experience (general and specific feature)	(2) Important
VMH3	Cultural learning	Create	[52]	Eliminate (heuristic not related to national park websites)	–
N2	The website should present information that generates interest in the users. The user should understand how to use sections of the website.	–	–	Useful and interesting content (specific feature)	(3) Critical

Table 14
Set of usability/UX heuristics for national park websites.

ID	Name
NPH1	Visibility of system
NPH2	Multimedia resources
NPH3	Information of interest
NPH4	Match between system and the real world
NPH5	User control and freedom
NPH6	Consistency and standards
NPH7	Information credibility
NPH8	Error prevention
NPH9	Minimize the user's memory load
NPH10	Flexibility and efficiency of use
NPH11	Aesthetic and minimalist design
NPH12	Help the user recover from errors
HPN13	Help and documentation
HPN14	Asynchronous interaction

Practitioners are experts conducting heuristic evaluations and they know how to correctly use a set of heuristics to find usability/UX problems. The activities to perform are as follows:

1. Select participants for Expert Judgment: We recommend including Researchers and Practitioners for expert judgment. This is because it is possible that an expert in HCI (Researcher) is not an expert in conducting a heuristic evaluation (Practitioner). It is important to have both points of view to validate the heuristics. It is important to mention that an expert can be a Researcher and a Practitioner at the same time. If possible, it would also be useful to include experts in the specific application domain. This is because it is possible that an expert in HCI is not expert in the specific application domain. It is important to have both points of view to verify that all the specific features of the application are covered and are evaluated with the newly proposed set of heuristics. In addition, we recommend that at least three experts validate the new set of heuristics.
2. Perform the expert judgment through survey: We propose using a questionnaire that assesses evaluators' perceptions of the new set of usability/UX heuristics, concerning 4 dimensions and 3 questions:
 - D1 – Utility: How useful the usability/UX heuristic is.
 - D2 – Clarity: How clear the usability/UX heuristic is.
 - D3 – Ease of use: How easy it was to associate identified problems with the usability/UX heuristic.
 - D4 – Necessity of an additional checklist: How necessary is it to complement the usability/UX heuristic with a checklist.
 - Q1 – Easiness: How easy it was to perform the heuristic evaluation, based on this set of usability/UX heuristics?
 - Q2 – Intention: Would you use the same set of usability/UX heuristics when evaluating similar software products in the future?
 - Q3 – Completeness: Do you think the set of usability/UX heuristics covers all usability aspects for this type of software product?

The set of usability/UX heuristics is rated globally through the 3 questions, but each heuristic is rated separately, on each of the 4 dimensions. All experts are asked to rate each usability heuristic, on each of the 4 dimensions, using a 5-point Likert scale. The value 1 indicates that the heuristic does not comply with the dimension and the value 5 indicates that the heuristic fully complies with the evaluated dimension. The 3 questions aim at evaluating the evaluators' overall perceptions; responses are also based on a 5-point Likert scale. The results of applying the survey that we propose are documented in [53–55].

6.7.3. Validation through user test

The user test is a complementary validation to the two validations that were previously discussed. Depending on the results that were obtained in the heuristic evaluation, it may be necessary to analyze these results from the perspective of the user. The user test can be useful for performing the following tasks:

1. Check if the usability/UX problems that were identified in the heuristic evaluation using the new set of heuristics are usability/UX problems for users.
2. Determine why the experimental group did not detect some of the problems that were detected by the control group. The tasks to be performed in the user test should involve the problems that were not detected by the experimental group, to determine why they were not detected.
3. Obtain the perceptions of users regarding a specific topic (for example, what features they consider relevant for the specific application domain).

6.8. Step 8: Refinement stage

[Table 16](#) shows some examples about how to document the heuristics to be refined, eliminated and/or created based on the set of usability/UX heuristics for national park websites. [Appendix K](#) shows the BPMN diagram for Step 8: Refinement stage.

7. Validating the methodology

We performed the methodology validation in two iterations. HCI experts and researchers participated in both iterations. Each participant was given the full methodology specification, which was divided into two parts: a brief version and a detailed version. The validation's objective was to obtain feedback from experts and researchers to improve the methodology.

7.1. First iteration: validation through expert opinion

In the first iteration, a validation was performed on an earlier version of the methodology. This first validation consisted of asking an expert and five researchers what they think about the proposed methodology. All the participants have experience performing heuristic evaluations and using sets of usability heuristics, so they can give their points of view regarding whether the methodology is correctly specified, whether it is useful, and what elements they would add or remove, among others.

In this first validation, the following were obtained:

1. A review of the methodology by an HCI expert.
2. Comments and suggestions from five researchers who have used (and continue to use) the methodology to create new sets of heuristics.

Based on the obtained feedback, the methodology was refined. All the comments and recommendations were taken into account to improve the methodology. The improvements that were suggested by both the expert and the researchers and the actions that were taken are presented in [Table 17](#).

7.2. Second iteration: validation through expert judgment

The methodology was recently applied in several case studies: on-line travel agencies, mobile apps for on-line travel agencies, parliamentary websites, websites for national parks, social networks, secure software systems, and 3D First Person Shooter (FPS) videogames. Sets of specific (usability/UX/playability) heuristics were developed in several iterations by 9 usability/UX experts. None of them had previous experience in developing sets of heuristics, but they were all familiar with the methodology that was proposed by Rusu et al. [9].

Based on the quantitative and qualitative results that were obtained, the methodology has been improved. Some of the comments and suggestions have been implemented, while others were not considered. The explanation of the changes that were made (and those that were not) is detailed below.

7.2.1. Survey

In the second iteration, we conducted a survey of 9 experts. They were asked to evaluate each step of the methodology separately, using a Likert

scale of 5 levels (1 – worst to 5 – best) in 4 dimensions (D1, D2, D3, and D4):

- (D1) *Utility* – how useful the step is considered.
- (D2) *Clarity* – how clearly the step's specification is perceived.
- (D3) *Ease of Use* – how easy it is to perform the step.
- (D4) *Necessity for More Details* – how necessary it is to provide more details for the step's specification.

The survey included 5 questions (Q1, Q2, Q3, Q4, and Q5) that rate the methodology globally using the same type of Likert scale as for dimensions D1 to D4:

- (Q1) *Easiness* – how easy was it to develop the set of heuristics using the methodology?
 - (Q2) *Overall Utility* – does the methodology facilitate the development of new heuristics?
 - (Q3) *Intention of Future Use* – how probable is it that the methodology will be used when developing future sets of heuristics?
 - (Q4) *Completeness* – does the methodology cover all aspects of the development of new heuristics?
 - (Q5) *Additional Graphics* – is a more-detailed graphical specification of the methodology required for better understanding?
- Additionally, 4 open questions were asked to obtain qualitative comments:
- (C1) – What specific aspects of the methodology were most difficult to apply in practice?
 - (C2) – What elements, tasks or explanations do you think are missing and should be included?
 - (C3) – What elements, tasks or explanations do you think are overemphasized (too many details are given)?
 - (C4) – Do you think the methodology should include any additional steps? If so, for what purpose?

7.2.2. Quantitative results

[Table 18](#) summarizes the results of the survey for dimensions D1 to D4. Descriptive statistics are analyzed below.

(D1) *Utility*. The average utility is high (4.68). Step 6 (Specification stage) is considered the most useful; all experts evaluated its utility with the highest score (5). Step 2 (Experimental stage) is considered “less” useful; however, its average utility is quite high (4.0). The standard deviation is relatively low (and the lowest among the 4 dimensions); it ranges from 0 (Step 6) to 0.73 (Step 2). The methodology's perceived utility is remarkably high; the lowest perceived utility is still high (Step 2).

(D2) *Clarity*. The average specification clarity is also high (4.40); it ranges from 3.89 (Step 4, Correlational stage) to 4.89 (Step 1, Exploratory stage). The standard deviation ranges from 0.33 (Step 1) to 1.05 (Step 2, Experimental stage, and Step 4, Correlational stage). The methodology's perceived clarity is high. Step 4 is perceived as the least clear step. Thus, the specification of this stage has been improved.

(D3) *Ease of Use*. The average ease of use is reasonable (3.61); it ranges from 3.11 (Step 4, Correlational stage) to 4.33 (Step 1, Exploratory stage). The lowest standard deviation occurs for Step 3 (Descriptive stage). For all other steps, the standard deviation is relatively high (ranging from 0.73 to 0.93). Step 4 is perceived as the most difficult to perform.

(D4) *Necessity for More Details*. The necessity for more details is low (average 2.64); its lowest value corresponds to Step 1 (Exploratory stage) and the highest values correspond to Step 4 (Correlational stage) and Step 5 (Selection stage). The standard deviation is high, from 0.93 (Step 1, Exploratory stage) to 1.80 (Step 4, Correlational stage). The experts' opinions on dimension D4 are heterogeneous.

The experts' opinions are approximately homogeneous for all dimensions, except D4. Step 4 (Correlational stage) is perceived as the least clear and most difficult to perform step, and its specification is perceived to require more details (D4). For this reason, Step 4 has been modified to improve its definition and ease of use. Step 1 (Exploratory stage) scores the best in terms of perception in dimensions D2, D3, and D4; its specification

Table 15

HPN2: multimedia resources heuristic.

ID	NPH2
Priority	(3) Critical
Name	Multimedia resources
Definition	The website should be attractive to the user by providing a virtual experience when navigating through different multimedia resources.
Explanation	Multimedia resources such as images, videos and audios must be present on the national park websites, display the information in different ways and complement one another.
Application feature	Specific feature: multimedia resources.
	The heuristic aims at evaluating the existence of multimedia resources, which contribute to the website and improve the virtual experience when navigating through it.
Examples	Several images may document the use of multimedia resources.
Benefits	The satisfaction of the user increases when he or she finds different multimedia resources that complement one another and improve the virtual experience.
Checklist	<ol style="list-style-type: none"> 1. All multimedia resources are available for viewing. 2. The website provides images, videos and/or interactive audios that complement one another and convey the contents of the national parks. 3. The website uses multimedia resources to capture the user's attention and make the user want to navigate the physical parks. 4. The website offers complementary options that work correctly, such as print, zoom, and download.
Usability attributes	Satisfaction
UX factors	Findable, Desirable, Valuable
Set of heuristics related	Heuristics for virtual museums [52].

Table 16

Heuristics to create, refine and/or eliminate (why and how to do it).

Heuristic	Problem	Action
NPH2 (<i>Content display</i>)	The definition of NPH2 is similar to that of NPH15 (<i>Aesthetic and minimalist design</i>).	Eliminate. NPH15 covers the same feature as NPH2.
NPH15 (<i>Aesthetic and minimalist design</i>)	The definition of NPH15 is similar to that of NPH2 (<i>Content display</i>).	Adapt. NPH15 should be complemented with the information that is detailed in NPH2 (<i>Content display</i>).
NPH5 (<i>Match between system and the real world</i>)	Half of the problems that were detected in the heuristic evaluation were incorrectly associated.	Adapt. The specification should be improved. In addition, the evaluation of multiple languages must be included.

is clear, additional details are not needed, and it is easy to perform.

The overall perception of the methodology (questions Q1–Q5) is presented in Table 19. We observe the following:

- On Easiness (Q1), it scored the lowest (average of 3.44) and had the most heterogeneous perceptions (standard deviation of 0.88); applying the methodology in practice is not a trivial task.
- On Overall Utility (Q2), it obtained the highest score (average of 4.89). The experts' opinions are remarkably positive; they think the methodology facilitated their work.
- The Intention of Future Use (Q3) score is remarkably high (average of 4.78); experts think it is very likely that the methodology will be used when developing new sets of heuristics.
- The methodology's Completeness (Q4) is perceived as reasonable (average of 3.78). Only one expert gave it a low rating (2); the other 8 experts gave it a high rating (4).
- On the need for Additional Graphics (Q5), an average score of 3.78 was obtained. The same expert that rated the methodology's Completeness as low (2) also rated as low (2) the need for Additional Graphics.

Even if the experts think that applying the methodology is not easy, they perceived it as useful and intend to use it in the future. The methodology's specification is perceived as fair, but the experts think that additional graphics would help.

As no assumption of normality could be made, the survey results were analyzed using nonparametric statistical tests. In all tests, $p \leq 0.05$ was used as the decision rule.

The Friedman test was performed to detect differences among experts' perceptions of the 8 steps of the methodology. The results (Table 20) indicate the following:

- The methodology's steps are not perceived as equally useful (D1), nor as equally easy to perform (D3).

- The steps' specifications are perceived as similarly clear (D2); the need for more detailed specifications (D4) is also perceived similarly for all steps.

The Spearman ρ test was performed to check the hypothesis:

$H_0: \rho = 0$, the dimensions/questions Dm/Qn and D/Qn are independent.

$H_1: \rho \neq 0$, the dimensions/questions D/Qm and D/Qn are dependent.

Table 21 shows the following:

- Most correlations are non-significant at the 0.05 level (marked as "N").
- There is a very strong significant correlation between D2 – Clarity and Q1 – Easiness; when steps' specifications are perceived as clear, the methodology is perceived as easy to apply in practice.
- There is also a very strong significant correlation between D3 - Ease of Use and Q1 – Easiness; when the methodology's steps are perceived as easy to perform, the methodology is perceived as easy to apply.
- There is a strong significant correlation between D3 – Ease of Use and Q3 – Intention of Future Use; when the methodology's steps are perceived as easy to perform, experts express their intention to use the methodology in the future.
- There is an unexpected very strong significant correlation between Q4 – Completeness and Q5 – Additional Graphics. When experts think that the methodology covers all aspects of the development of new heuristics, they still feel the need for more-detailed graphic specifications.

The few significant correlations that occur are (very) strong. Most of them are expected. The only unexpected correlation is between Q4 and Q5: when the methodology is perceived as complete, the experts still feel that additional graphics could help facilitate understanding. The perceived need for additional graphics could be due to the experts' backgrounds (engineering, computer science).

Table 17

Improvements that were suggested by both the expert and the researchers.

Step	Comment	Action
Methodology in general	It was not suggested to add any stage to or eliminate any stage from the methodology. The expert suggested indicating at each stage the inputs and outputs. Although these are shown in the BPMN diagrams (as input and output objects), it is recommended that this information also be detailed in the description of each stage.	We clearly included the inputs ("What do I need to get started?") and outputs ("What is obtained?") for each stage.
Step 1: Exploratory stage	The expert suggested that at this stage, the information about usability/UX attributes must be collected. The new set of heuristics must evaluate both features of the specific application domain and attributes of usability/UX.	This component is very important for the creation of heuristics, so it has been added at this stage. In addition, it is fundamental to perform Step 4: Correlation stage.
Step 2: Experimental stage	Both the expert and the researchers indicated that this stage is not understood very well.	We improved the description of step 2 and we clearly explained its purpose: analyze and interpret previous experimental data to obtain useful information for creating the heuristics. In addition, we clearly explained that if there is time and evaluators are available, it is possible perform additional experiments. Otherwise, it is not recommended to perform them.
Step 3: Descriptive stage	The expert recommended defining a mechanism for prioritizing the information that was collected in Step 1. The expert proposed creating a "relevance scale" for the information.	We included a three-level scale for prioritizing the information that is collected in Step 3.
Step 4: Correlational stage	The expert advised explaining clearly and with examples the objective of this stage, since the definition is confusing.	We improved the step specification. This step consists of matching among features, usability/UX attributes, and existing heuristics. We also included examples to explain the matching process.
Step 5: Selection stage	Both the expert and the researchers indicated that there is a lack of explanation on how to document the selected heuristics. Although a standard table is proposed for documenting the selection of heuristics, with which information to complete the table and what to do in particular situations should be clearly explained.	We improved the step specification as follows: – We explained how to document deleted heuristics (and what information to include). – We indicated that it is possible to combine two or more heuristics into a single heuristic. – We clearly indicated that it is important to eliminate redundancy (multiple sets that have the same heuristics should not be included in the selection process).
Step 6: Specification stage	The expert suggested adding two new elements to the template for specifying heuristics: – Application feature (justification): The aspect or feature of the specific application domain that each heuristic evaluates should be indicated. – Usability/UX attribute: The usability/UX attribute that each heuristic evaluates should be indicated, to cover both aspects of the specific application domain and usability/UX attributes in the definition of the heuristic.	We added two new elements to the template for specifying heuristics: Application feature and Usability/UX attribute.
Step 7: Validation stage	The expert recommended clearly indicating that it is important to perform more than one validation to obtain more feedback, so that it will be possible to compare and/or complement the results. For "Validation through Heuristic Evaluation", the expert suggested the following: – Use the term "control heuristics" rather than "traditional heuristics" since it is possible to create new sets based not only on Nielsen heuristics but also on other existing sets of heuristics. – Explain the scales of severity, frequency and criticality that are used. In addition, clearly indicate that these are suggested scales and that the researcher can decide which scales to use and, based on them, adapt the presented formulas. For "Validation through Expert Judgment", the expert proposed the following: – Explain clearly the types of experts that it is possible to survey and how to select them. – Explain how to analyze and interpret the results that are obtained in the surveys.	All the suggestions were taken into account. We improved the step specification.
Step 8: Refinement stage	The expert recommended including the feedback that is obtained from Step 7: Validation stage, to clearly determine what to refine and what steps should be repeated (if it is decided to iterate).	We clearly show the inputs for step 8 (outputs of step 7).

As dimensions D1, D2, D3 and D4 have been evaluated for each step of the methodology separately, we also performed the Spearman ρ test for each step. Only three significant correlations occurred:

- For Step 4 (Correlational stage), a strong negative correlation (-0.771) between D2 – Clarity and D4 – Necessity for More Details; when Step 4 is perceived as less clear, the experts think its specification requires more details.
- For Step 6 (Specification stage), a strong correlation (0.719) between D2 – Clarity and D3 – Ease of Use; when Step 6 is perceived as clear, it is also perceived as easy to perform.
- For Step 7 (Validation stage), a very strong correlation (0.804)

between D1 – Utility and D2 – Clarity; when Step 7 is perceived as clear, it is also perceived as useful.

7.2.3. Qualitative results

When asked about specific aspects of the methodology that were most difficult to apply in practice (C1), the experts indicated the following:

- It is difficult and time-consuming to match usability/UX attributes and domain-specific characteristics (Step 4, Correlational stage; 5 comments). If done inadequately, the set of heuristics will not be suitable for their purpose. Step 4 has been clarified. The methodology that is presented in Section 5 is the final version after this refinement.

Table 18

Survey results for dimensions D1, D2, D3 and D4.

		Step 1 – Exploratory stage	Step 2 – Experimental stage	Step 3 – Descriptive stage	Step 4 – Correlational stage	Step 5 – Selection stage	Step 6 – Specification stage	Step 7 – Validation stage	Step 8 – Refinement stage	Mean
D1 – Utility	Mean	4.89	4.00	4.67	4.44	4.89	5.00	4.78	4.78	4.68
	Std. dev.	0.33	0.71	0.50	0.73	0.33	0.00	0.44	0.44	
D2 – Clarity	Mean	4.89	4.11	4.44	3.89	4.22	4.56	4.56	4.56	4.40
	Std. dev.	0.33	1.05	0.53	1.05	0.97	0.73	0.73	0.73	
D3 – Ease of Use	Mean	4.33	3.44	4.00	3.11	3.33	3.78	3.44	3.44	3.61
	Std. dev.	0.87	0.88	0.50	0.93	0.87	0.83	0.73	0.73	
D4 – Necessity for More Details	Mean	2.11	2.56	2.44	3.00	3.00	2.67	2.44	2.89	2.64
	Std. dev.	0.93	1.33	1.42	1.80	1.00	1.22	1.24	1.17	

Table 19

Survey results for questions Q1, Q2, Q3, Q4 and Q5.

	Q1 – Easiness	Q2 – Overall utility	Q3 – Intention of future use	Q4 – Completeness	Q5 – Additional graphics
Mean	3.44	4.89	4.78	3.78	3.78
Std. dev.	0.88	0.33	0.44	0.67	0.67

Table 20

Friedman test results.

Dimensions	D1 – Utility	D2 – Clarity	D3 – Ease of use	D4 – Necessity for more details
p-Value	0.040	0.148	0.024	0.171

- Heuristic specification (Step 6, Specification stage) has to be performed carefully (2 comments). It requires details, precision, time, reviews, and (independent) validations. If not, the heuristics will be difficult to understand and apply in practice.
- Performing usability tests, as suggested in Step 7 (Validation stage), requires substantial time and resources (1 comment).
- When performing Step 5 (Selection stage), it was difficult to determine what aspect(s) a heuristic covers (1 comment).
- Step 1 (Exploratory stage) requires time and substantial information for its implementation (1 comment). However, the expert acknowledges that the step is critical.

When asked about what other elements, tasks or explanations the methodology should include (C2), the experts made a few comments. Table 22 shows the experts' comments, whether the suggestion has been taken into account, why and what changes have been made (when applicable).

When asked whether any elements, tasks or explanations are overemphasized in the methodology's specification by providing too many details (C3), most of the experts indicate that this is not the case. They also think that specifying the methodology in two forms (a brief form and a detailed form) is helpful. Only two comments mention that many details

are given in some cases (Step 1: Exploratory stage, Step 2: Experimental stage, and Step 7: Validation stage); however, the two experts also admit that the nature of those steps requires detailed specifications.

When asked if the methodology should include any additional steps (C4), all experts agree that the methodology is complete.

The experts' comments are positive. They think the methodology is complete and offers the appropriate level of detail. They also highlight that offering two levels of details (a synthesized specification and a detailed specification) help researchers with different levels of expertise (in developing heuristics) better understand the methodology. Some suggestions were made and have already been implemented. The methodology was refined based on the experts' opinions, especially Step 4 (Correlational stage).

8. Conclusions

Heuristic evaluation is one of the most popular inspection methods [43]. Several sets of usability/UX heuristics have been developed for specific application domains, for evaluating both general and specific features. The heuristics should be well-designed, easy to use, and make it possible to identify domain-related usability/UX problems.

Following a methodology when specifying and validating heuristics makes development easier since it provides well-defined steps and proposes methods for validating new heuristics. If no methodology for developing heuristics is used, the created heuristics may be (1) difficult to understand, (2) difficult to use, and/or (3) not effective in evaluating the usability/UX of a specific application.

The methodology that was proposed by Rusu et al. [9] has been used to develop several sets of heuristics for specific applications. However, this methodology has deficiencies in explaining stages and generates confusion on how to iterate and apply it properly. Therefore, we develop a new methodology by making several changes to methodology that was proposed by Rusu et al. [9], namely, adding new steps, definitions and diagrams and improving the specification of the methodology. The new methodology presents tables that summarize the inputs, outputs and activities to be performed at each stage. We also included BPMN diagrams for each stage.

Two validations were performed to validate and refine the methodology. Most of the comments and suggestions from the experts were

Table 21Spearman ρ test for D1, D2, D3, D4, and Q1, Q2, Q3, Q4, Q5.

D1 – Utility	D2 – Clarity	D3 – Ease of use	D4 – Necessity for more details	Q1 – Easiness	Q2 – Overall utility	Q3 – Intention of future use	Q4 – Completeness	Q5 – Additional Graphics
D1 1	N	N	N	N	N	N	N	N
D2	1	N	N	0.818	N	N	N	N
D3		1	N	0.829	N	0.687	N	N
D4			1	N	N	N	N	N
Q1				1	N	N	N	N
Q2					1	N	N	N
Q3						1	N	N
Q4							1	1.000
Q5								1

Table 22

Comments from experts.

Comment	Is the suggestion taken into account?	Reason/action
1. Guidelines on how to interpret user test results (performed in Step 2: Experimental stage and/or Step 7: Validation stage) could be helpful (1 comment).	No	Recommendations on how the results of the user tests that were performed in Step 7 should influence Step 8 (Refinement stage) could reduce the impact of the experts' subjective judgements. This suggestion has not been taken into account since the results that were obtained in the user tests depend on the type of test that was applied and the design of the test. However, it is not discarded and will be considered when adding guidelines in the future.
2. Specifying more detailed subtasks and how they could be iterated may be helpful (1 comment).	No	Most experts agree that the methodology has an appropriate level of detail and adding more elements could make it more difficult to use. Therefore, the suggestion will not be considered.
3. Highlighting the inputs/outputs (outcomes) of each step could be helpful (1 comment).	Yes	The recommendation has been taken into account and the inputs and outputs of each stage have been clearly highlighted, in both the brief version and the detailed version of the methodology.
4. Examples of how to match usability/UX attributes, domain-related application features, and heuristics could help (1 comment).	Yes	Specifying the degree of a heuristic's coverage (non-covered, partially covered, or covered) could also be helpful. This suggestion has been taken into account. The specification of Step 4 has been complemented by adding clear examples on how to match usability/UX attributes, domain-related application's features, and heuristics.
5. The specification of Step 4 (Correlational stage) may be improved. Specification of the relationship between Step 4 and Step 5 (Selection stage) would probably facilitate understanding of Step 6 (Specification stage) (1 comment).	Yes	This recommendation has been taken into account and the specifications of Step 4 and 5 have been improved.
6. Examples of how two different heuristics could be merged in Step 5 (Selection stage) could probably be of help (1 comment).	No	The comment was made only once. We think that specific examples would only overemphasize the specification.

taken into consideration. In the first validation (first iteration), all stages were refined, and new elements were added to the specification of the methodology. In the second validation (second iteration), changes were made mainly to the specification of Step 4 (Correlational stage).

Based on the results that were obtained from the survey (second validation), we concluded that the experts perceived the methodology to be useful and intend to use it in the future. The methodology's specification is perceived as fair, but the experts think that additional graphics would help. Therefore, we added the BPMN diagrams to facilitate better understanding of the methodology. The experts stated that the stages are explained in sufficient detail and that it is not necessary to add a new stage. In addition, they indicated that presenting both brief and detailed versions of the methodology is useful.

The methodology was recently applied in several case studies to develop new sets of usability/UX heuristics for specific application domains. We conclude that the methodology is effective since it has been used to create new sets of heuristics (a final "product" is obtained). Moreover, it is perceived as useful by the experts who have used it.

The methodology is applicable for developing heuristics for UX-related aspects, such as playability, communicability, and learnability. We also consider the methodology to have great potential in the

development of instruments for other quality attributes that require an instrument of heuristic type or guidelines, such as security and adaptability. It has already been applied to develop playability and mixed usability-security heuristics.

In future work, we intend to study how the methodology can be used to develop heuristics for the evaluation of quality attributes other than usability. The methodology's efficiency could be evaluated. However, efficiency criteria should be defined first.

Acknowledgments

The authors would like to thank all the participants (experts and researchers) who were involved in the experiments for this study, especially the members of the "UseCV" Research Group in Human–Computer Interaction. We would like to thank the experts who developed the heuristics for national park websites for sharing their work. This work was supported by Escuela de Ingeniería Informática (School of Informatics Engineering) of the Pontificia Universidad Católica de Valparaíso – Chile. Daniela Quiñones has been granted the "INF-PUCV" and "Postgrado PUCV 2017" Graduate Scholarship.

Appendix A. BPMN diagram for Step 1: Exploratory stage

Fig. 5.

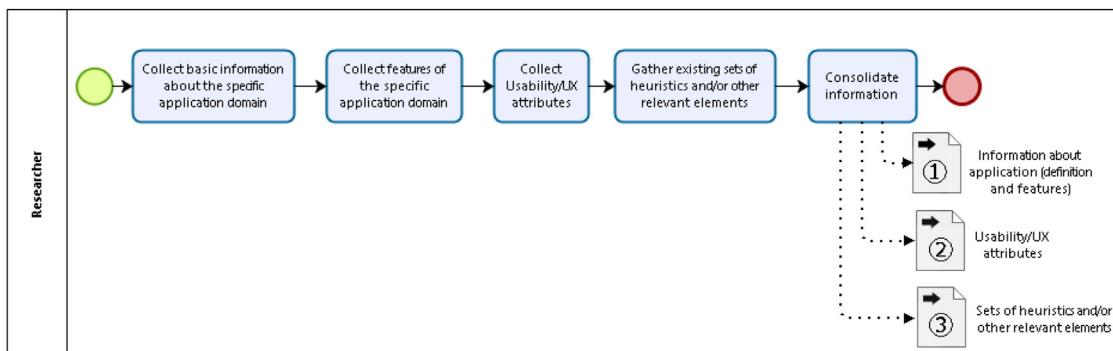


Fig. 5. BPMN diagram – exploratory stage.

Appendix B. BPMN diagram for Step 2: Experimental stage

Fig. 6.

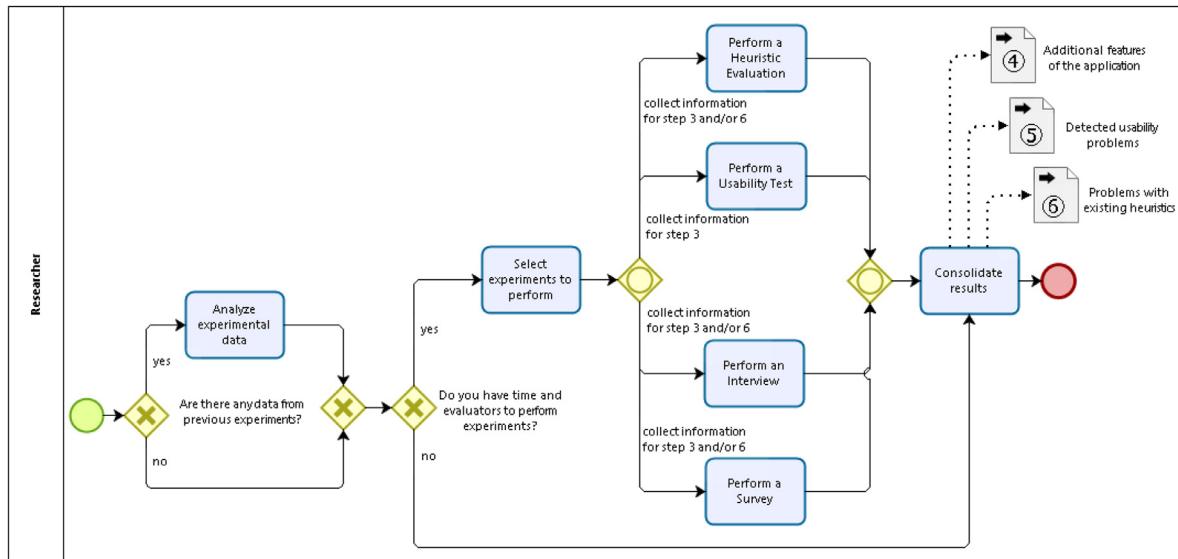


Fig. 6. BPMN diagram – experimental stage.

Appendix C. BPMN diagram for Step 3: Descriptive stage

Fig. 7.

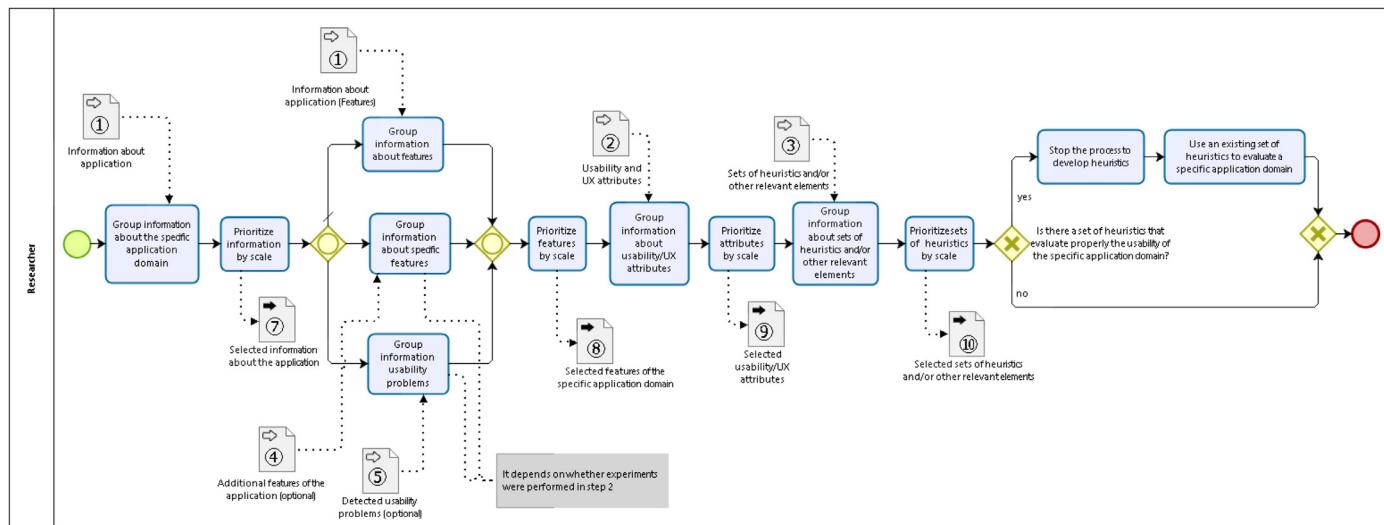


Fig. 7. BPMN diagram – descriptive stage.

Appendix D. BPMN diagram for Step 4: Correlational stage

Fig. 8.

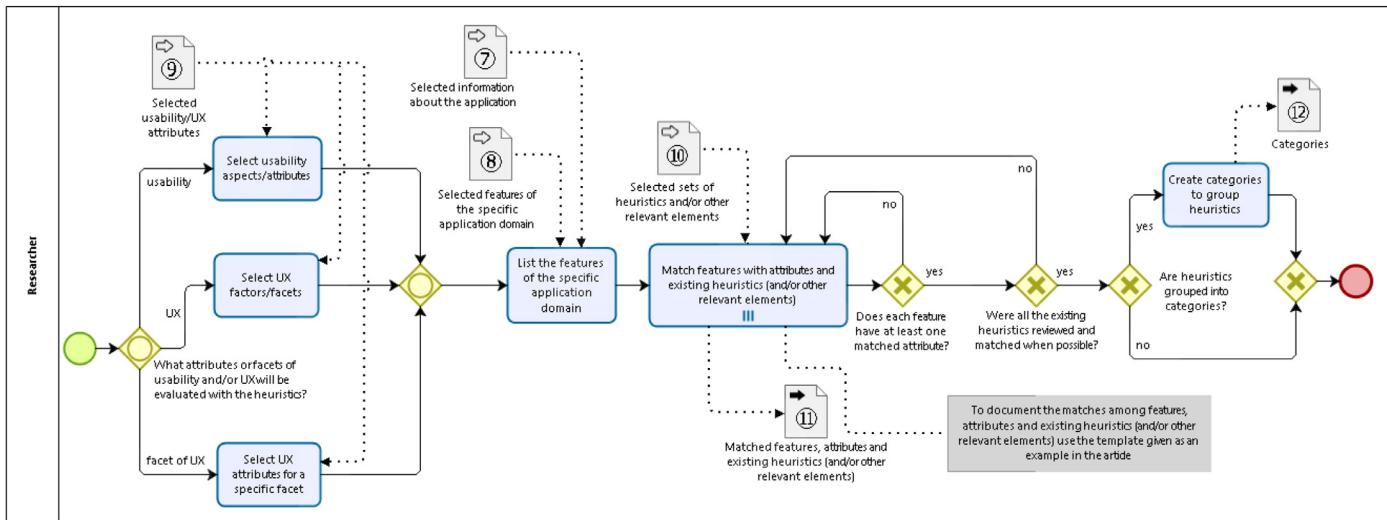


Fig. 8. BPMN diagram – correlational stage.

Appendix E. BPMN diagram for Step 5: Selection stage

Fig. 9.

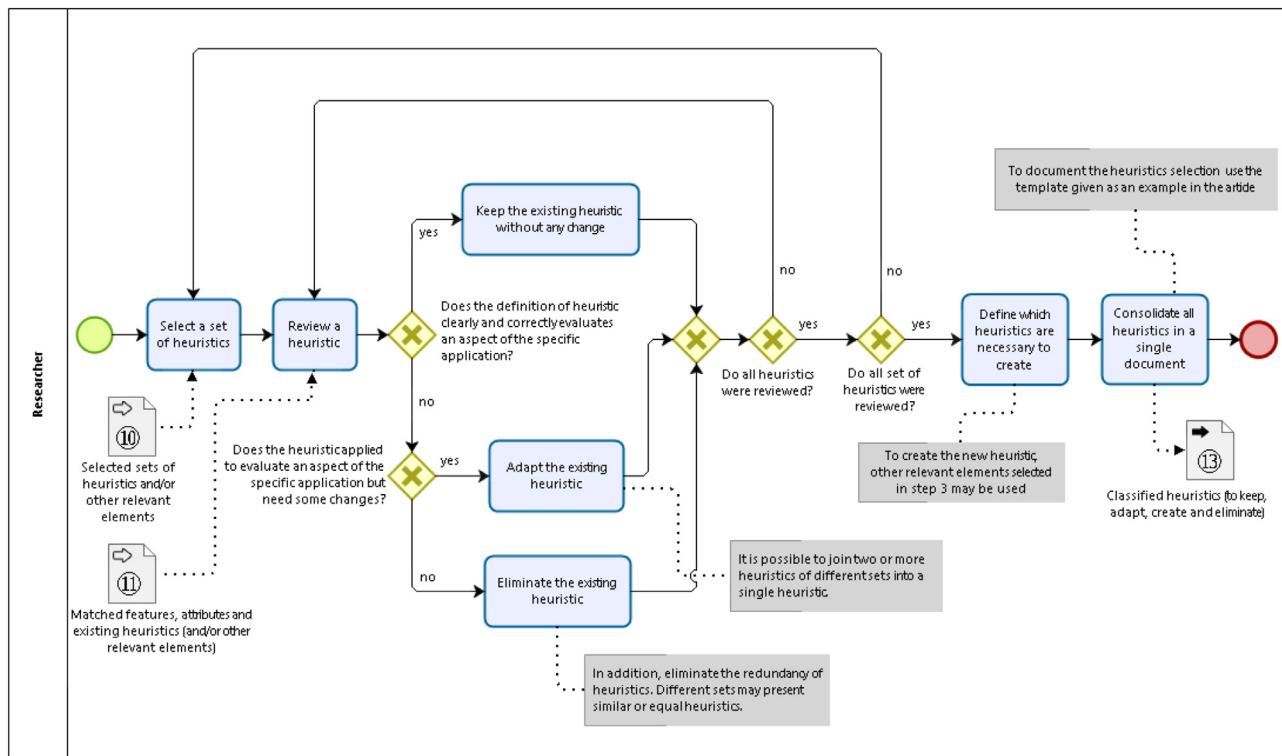


Fig. 9. BPMN diagram – selection stage.

Appendix F. BPMN diagram for Step 6: Specification stage

Fig. 10.

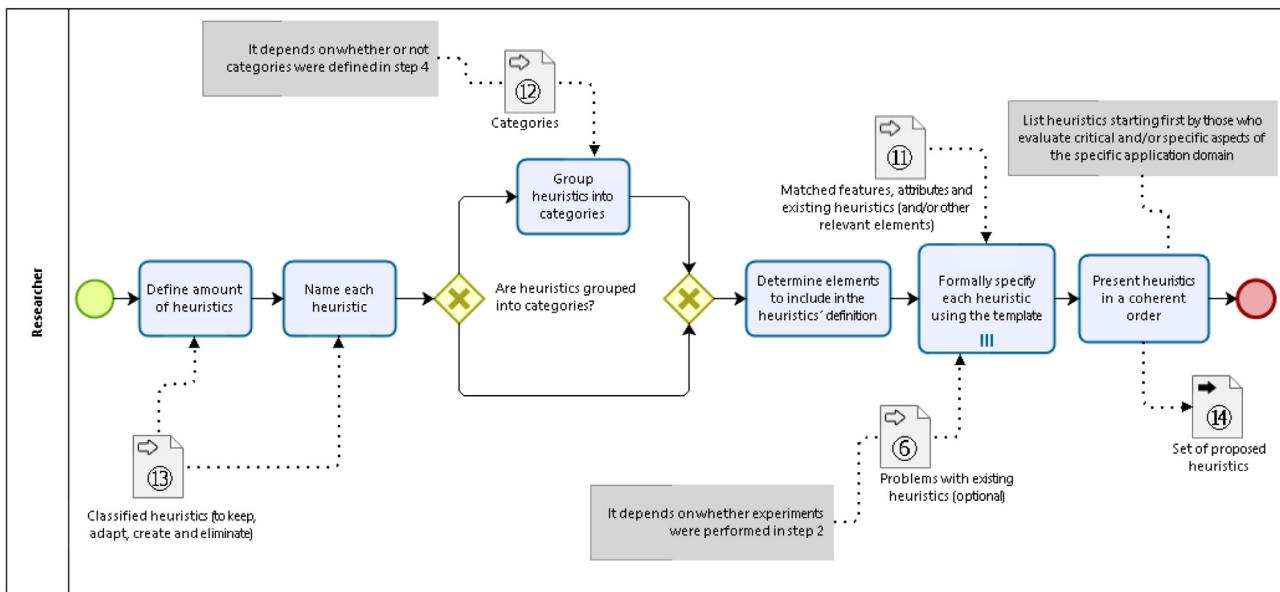


Fig. 10. BPMN diagram – specification stage.

Appendix G. BPMN diagram for Step 7: Validation stage

Fig. 11.

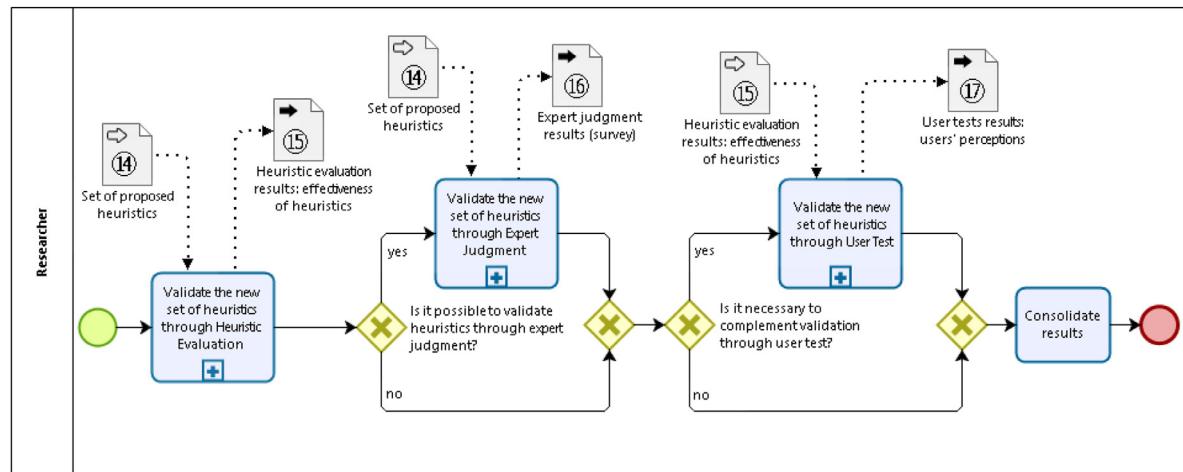


Fig. 11. BPMN diagram – validation stage.

Appendix H. BPMN diagram for validation through heuristic evaluation

Fig. 12.

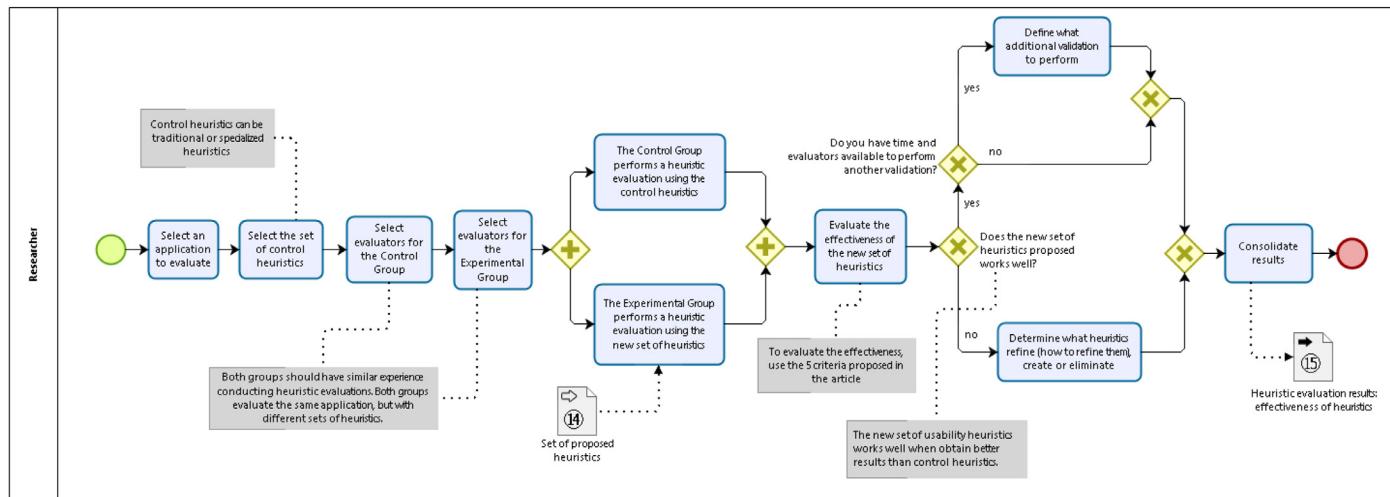


Fig. 12. BPMN diagram – validation through heuristic evaluation.

Appendix I. BPMN diagram for validation through expert judgment

Fig. 13.

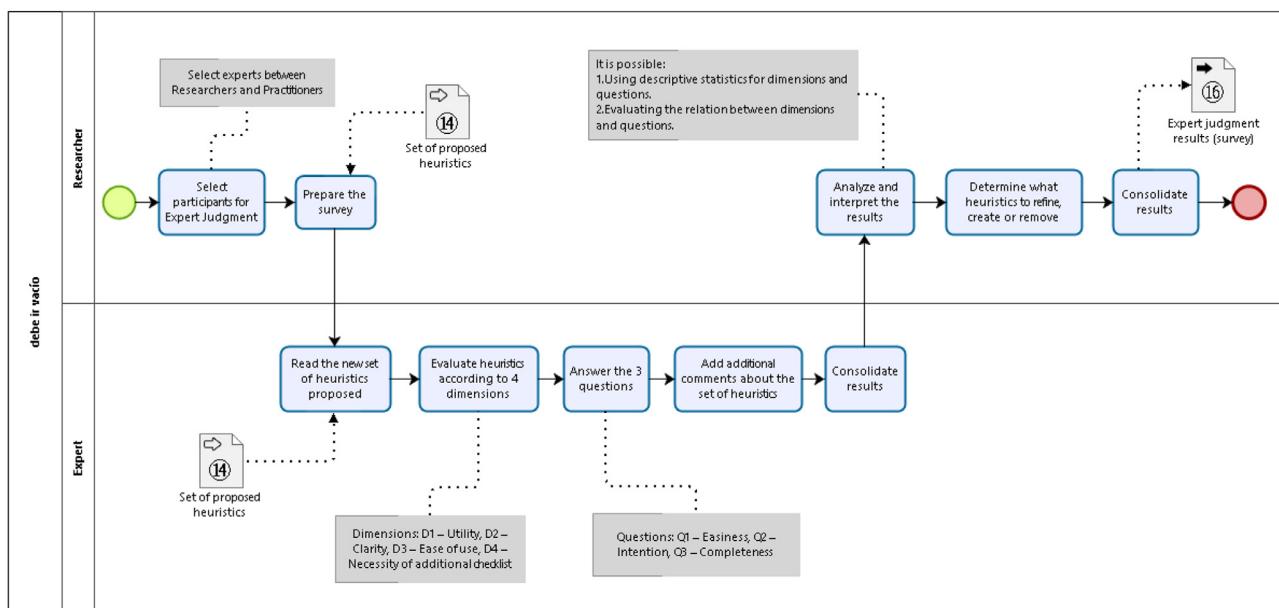


Fig. 13. BPMN diagram – validation through expert judgment.

Appendix J. BPMN diagram for validation through user test

Fig. 14.

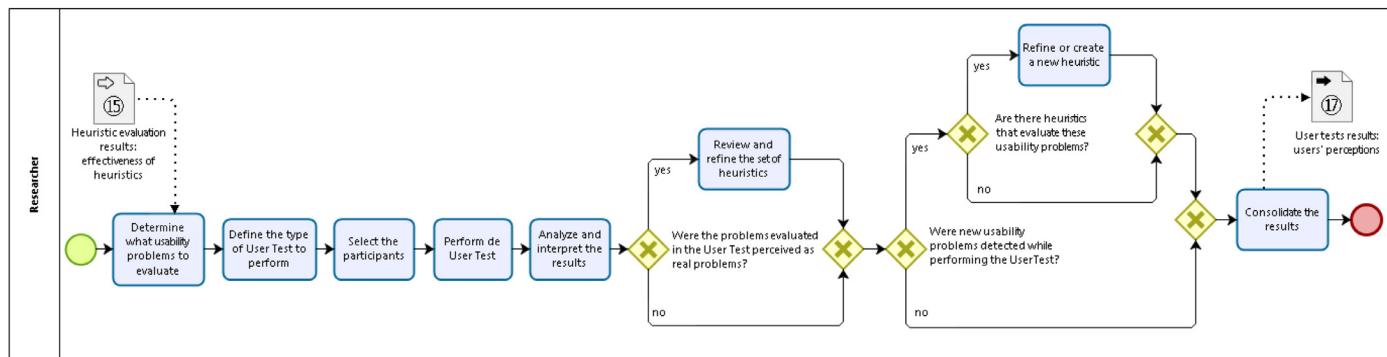


Fig. 14. BPMN diagram – validation through user test.

Appendix K. BPMN diagram for Step 8: Refinement stage

Fig. 15.

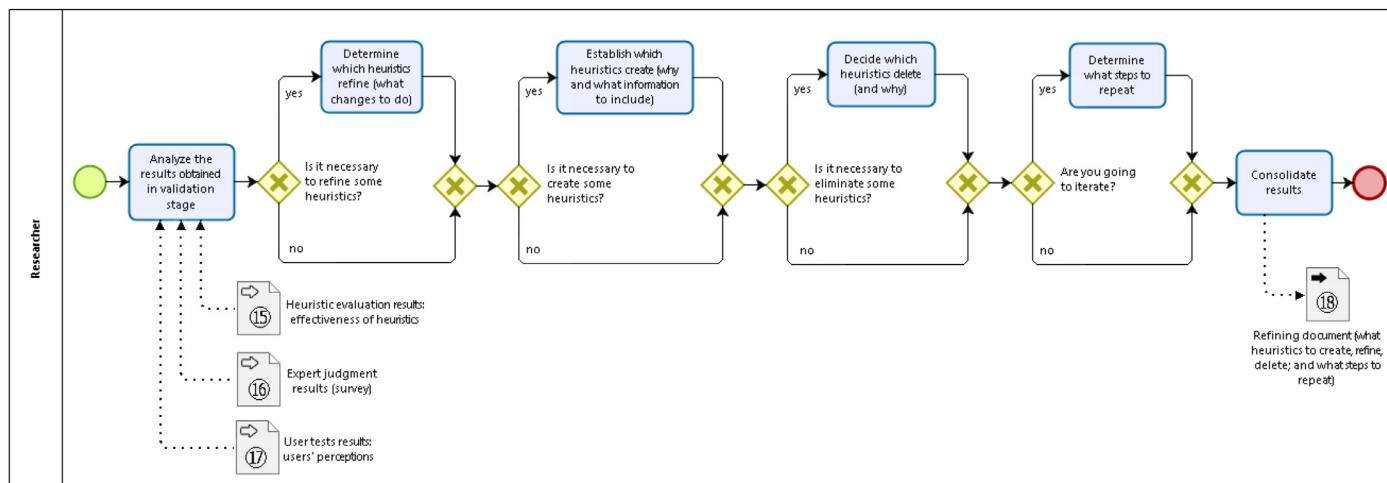
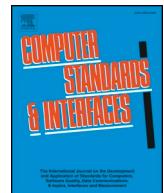


Fig. 15. BPMN diagram – refinement stage.

References

- [1] N. Bevan, J. Carter, S. Harker, ISO 9241-11 revised: what have we learnt about usability since 1998? in: M. Kurosu (Ed.), Human-Computer Interaction: Design and Evaluation, Vol. 9169 Springer, 2015, pp. 143–151 Lecture Notes in Computer Science.
- [2] ISO 9241-210, Ergonomics of Human-system Interaction — Part 210: Human-centred Design for Interactive Systems, International Organization for Standardization, 2010.
- [3] H. Hartson, T. Andre, R. Williges, Criteria for evaluating usability evaluation methods, Int. J. Hum.-Comput. Interact. 13 (4) (2001) 373–410.
- [4] J. Scholtz, Usability Evaluation, National Institute of Standards and Technology, 2004.
- [5] D. Quiñones, C. Rusu, How to develop usability heuristics: a systematic literature review, Comput. Stand. Interfaces 53 (2017) 89–122.
- [6] P. Jaferian, K. Hawkey, A. Sotirakopoulos, M. Velez-Rojas, K. Beznosov, Heuristics for evaluating IT security management tools, Hum.–Comput. Interaction 29 (4) (2014) 311–350.
- [7] S. Hermawati, G. Lawson, Establishing usability heuristics for heuristics evaluation in a specific domain: is there a consensus? Appl. Ergon. 56 (2016) 34–51.
- [8] G. Sim, J.C. Read, G. Cockton, Evidence based design of heuristics for computer assisted assessment, in: T. Gross, J. Gulliksen, P. Kotze, L. Oestreicher, P. Palanque, R.O. Prates, M. Winckler (Eds.), Proceedings of Human-Computer Interaction INTERACT 2009, 2009, pp. 204–216.
- [9] C. Rusu, S. Roncagliolo, V. Rusu, C. Collazos, A methodology to establish usability heuristics, Proceedings of the Fourth International Conference on Advances in Computer-Human Interactions, ACHI2011, 2011, pp. 59–62.
- [10] D. Van Greunen, A. Yeratziotis, D. Pottas, A three-phase process to develop heuristics, Proceedings of the 13th ZA-WWW Conference, Johannesburg, 2011.
- [11] M. Hub, V. Čapková, Heuristic evaluation of usability of public administration portal, DEO, Narsingh, et al. Applied Computer Science, Proceedings of International Conference on Applied Computer Science, ACS, 2010.
- [12] F. Franklin, F. Breyer, J. Kelner, Heurísticas de usabilidade para sistemas colaborativos remotos de realidade aumentada, Proceedings of XVI Symposium on Virtual and Augmented Reality, SVR, 2014, pp. 53–62.
- [13] B. Lechner, A. Fröhling, S. Petter, H. Siy, The chicken and the pig: user involvement in developing usability heuristics, Proceedings of the Nineteenth Americas Conference on Information Systems, 2013.
- [14] S. Hermawati, G. Lawson, A user-centric methodology to establish usability heuristics for specific domains, Proceedings of the International Conference on Ergonomics & Human Factors, 2015, pp. 80–85.
- [15] R. Irostiza, C. Rusu, S. Roncagliolo, V. Rusu, C. Collazos, Developing SMASH: a set of SMArtphone's usability Heuristics, Comput. Stand. Interfaces 43 (2016) 40–52.
- [16] ISO 9241-11, Ergonomic Requirements for Office Work with Visual Display Terminals (VDT's) – Part 11: Guidance on Usability, International Organization for Standardization, Geneva, 1998.
- [17] N. Bevan, International standards for HCI and usability, Int. J. Hum. Comput. Stud. 55 (4) (2001) 533–552.
- [18] A. Fernández, E. Insfran, S. Abrahão, Usability evaluation methods for the web: a systematic mapping study, J. Inf. Softw. Technol. 53 (2011) 789–817.
- [19] J. Nielsen, Usability inspection methods, Proceedings of Conference on Human factors in computing systems, Boston, Massachusetts, United States, 1994.
- [20] J. Nielsen, R. Molich, Heuristic evaluation of user interfaces, Proceeding of Conference on Human factors in Computing Systems, SIGCHI '90, 1990, pp. 249–256.
- [21] J. Nielsen, Ten Usability Heuristics, <https://www.nngroup.com/articles/ten-usability-heuristics/> (accessed 20 October 2017).
- [22] A. Vermeeren, E. Lai-Chong, V. Roto, M. Obrist, J. Hoonhout, K. Väänänen-Vainio-Mattila, User experience evaluation methods: current state and development needs,

- Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries, New York, 2010, pp. 521–530.
- [23] A. Allam, A. Razak, H. Mohamed, User experience: challenges and opportunities, *J. Inf. Syst. Res. Innov.* 3 (2013) 28–36.
- [24] All About UX, Information for User Experience Professionals, <http://www.allaboutux.org/> (accessed 21 October 2017).
- [25] C. Rusu, S. Roncagliolo, G. Tapia, D. Hayvar, V. Rusu, D. Gorgan, Usability heuristics for grid computing applications, Proceedings of the Fourth International Conference on Advances in Computer-Human Interactions, ACHI, 2011, pp. 53–58.
- [26] C. Rusu, R. Muñoz, S. Roncagliolo, S. Rudloff, V. Rusu, A. Figueiroa, Usability heuristics for virtual worlds, Proceedings of the Third International Conference on Advances in Future Internet, IARIA, 2011, pp. 16–19.
- [27] A. Solano, C. Rusu, C. Collazos, S. Roncagliolo, J.L. Arciniegas, V. Rusu, Usability heuristics for interactive digital television, Proceedings of the Third International Conference on Advances in Future Internet, IARIA, 2011, pp. 60–63.
- [28] J. Díaz, C. Rusu, J. Pow-Sang, S. Roncagliolo, A cultural – oriented usability heuristics proposal, Proceedings of the 2013 Chilean Conference on Human Computer Interaction, ChileCHI '13, 2013, pp. 82–87.
- [29] J. Díaz, C. Rusu, C.A. Collazos, Experimental validation of a set of cultural-oriented usability heuristics: e-Commerce websites evaluation, *Comput. Stand. Interfaces* 50 (2017) 160–178.
- [30] R.X.E. de Almeida, S.B.L. Ferreira, D.S. da Silveira, M. Pimentel, R. Goldbach and A. T. Bessa, “Heurísticas de Usabilidade Orientadas às Redes Sociais”, in IV Encontro de Administração da Informação, 2013.
- [31] K.R. da, H. Rodrigues, C.A.C. Teixeira, V.P. de, A. Neris, Heuristics for assessing emotional response of viewers during the interaction with TV programs, in: M. Kurosu (Ed.), Human–Computer Interaction, Part I, HCII, Springer, 2014, pp. 577–588 LNCS 8510.
- [32] E.V. Neto, F.F.C. Campos, Evaluating the usability on multimodal interfaces: a case study on tablets applications, Design, User Experience, and Usability. Theories, Methods, and Tools for Designing the User Experience, Springer International Publishing, 2014, pp. 484–495.
- [33] R. Yáñez Gómez, D.C. Caballero, J.L. Sevillano, Heuristic evaluation on mobile interfaces: a new checklist, *Sci. World J.* (2014).
- [34] F. Paz, F.A. Paz, J.A. Pow-Sang, L. Collantes, Usability heuristics for transactional web sites, Proceedings of the 11th International Conference on Information Technology, 2014, pp. 627–628.
- [35] N. Gale, P. Mirza-Babaei, I. Pedersen, Heuristic guidelines for wearable augmented reality applications, Proceedings of the 2015 Annual Symposium on Computer–Human Interaction in Play, CHI PLAY '15, 2015, pp. 529–534.
- [36] L.C. Rocha, R.M. Andrade, A.L. Sampaio, V. Lelli, Heuristics to evaluate the usability of ubiquitous systems, Proceedings of International Conference on Distributed, Ambient, and Pervasive Interactions, Springer, 2017, pp. 120–141.
- [37] F. Sanz, R. Gálvez, C. Rusu, S. Roncagliolo, V. Rusu, C.A. Collazos, J.P. Cofré, A. Campos, D. Quiñones, A set of usability heuristics and design recommendations for u-learning applications, *Information Technology: New Generations*, Springer International Publishing, 2016, pp. 983–993.
- [38] A. Campos, C. Rusu, S. Roncagliolo, F. Sanz, R. Gálvez, D. Quiñones, Usability heuristics and design recommendations for driving simulators, *Information Technology: New Generations*, Springer International Publishing, 2016, pp. 1287–1290.
- [39] A. Yeratziotis, D. Pottas, D. Vvan Greunen, A usable security heuristic evaluation for the online health social networking paradigm, *Int. J. Hum.-Comput. Interaction* 28 (10) (2012) 678–694.
- [40] L. Kuparinen, J. Silvennoinen, H. Isomäki, Introducing usability heuristics for mobile map applications, Proceedings of the 26th International Cartographic Conference, 2013.
- [41] J.S. Mtebe, M.M. Kissaka, Heuristics for evaluating usability of learning management systems in Africa, Proceedings of IST-Africa Conference, IEEE, 2015, pp. 1–13.
- [42] G. Joyce, M. Lilley, Towards the development of usability heuristics for native smartphone mobile applications, in: A. Marcus (Ed.), Design, User Experience, and Usability. Theories, Methods, and Tools for Designing the User Experience, 8517 Springer International Publishing, 2014, pp. 465–474.
- [43] A. Anganes, M.S. Pfaff, J.L. Drury, C.M. O'Toole, The heuristic quality scale, *Interact. Comput.* 28 (5) (2016) 584–597.
- [44] F. Paz, F.A. Paz, J.A. Pow-Sang, L. Collantes, Usability heuristics for transactional web sites, Proceedings of the 11th International Conference on Information Technology, 2014, pp. 627–628.
- [45] F. Paz, F.A. Paz, J.A. Pow-Sang, Experimental case study of new usability heuristics, Design, User Experience, and Usability: Design Discourse, Springer International Publishing, 2015, pp. 212–223.
- [46] D. Quiñones, C. Rusu, S. Roncagliolo, Redefining usability heuristics for transactional web applications, Proceedings of the 11th International Conference on Information Technology: New Generations, ITNG, IEEE, 2014, pp. 260–265.
- [47] D. Quinones, C. Rusu, S. Roncagliolo, V. Rusu, C. Collazos, Developing usability heuristics: a formal or informal process? *IEEE Lat. Am. Trans.* 14 (7) (2016) 3400–3409.
- [48] D. Quiñones, A methodology to develop usability/user experience heuristics, Proceedings of the XVIII International Conference on Human Computer Interaction, 2017 Article No. 57.
- [49] M. Chinosi, A. Trombetta, BPMN: an introduction to the standard, *Comput. Stand. Interfaces* 34 (1) (2012) 124–134.
- [50] J. Nielsen, Usability 101: Introduction to Usability, 2012. <https://www.nngroup.com/articles/usability-101-introduction-to-usability/> (Accessed 20 October 2017).
- [51] P. Morville, User Experience Design, 2004. http://semanticstudios.com/user_experience_design/ (Accessed 26 October 2017).
- [52] N. Aguirre, Experiencia de Usuario en Museos Virtuales, Undergraduate Thesis Pontificia Universidad Católica de Valparaíso, Chile, 2015.
- [53] C. Rusu, V. Rusu, S. Roncagliolo, D. Quiñones, V.Z. Rusu, H.M. Fardoun, D.M. Alghazzawi, C.A. Collazos, Usability heuristics: reinventing the wheel? Proceedings of the International Conference on Social Computing and Social Media, Springer International Publishing, 2016, pp. 59–70.
- [54] V. Rusu, C. Rusu, D. Quiñones, S. Roncagliolo, C.A. Collazos, What happens when evaluating social media's usability? Proceedings of the International Conference on Social Computing and Social Media, Springer, 2017, pp. 117–126.
- [55] V. Rusu, C. Rusu, D. Guzmán, S. Roncagliolo, D. Quiñones, Online travel agencies as social media: analyzing customers' opinions, Proceedings of the International Conference on Social Computing and Social Media, Springer, 2017, pp. 200–209.



Applying a methodology to develop user eXperience heuristics

Daniela Quiñones*, Cristian Rusu

Pontificia Universidad Católica de Valparaíso, Valparaíso, Chile



ARTICLE INFO

Keywords:

Usability
User experience
Heuristic evaluation
Usability heuristics
User experience heuristics
Methodology

ABSTRACT

A heuristic evaluation method allows the evaluation of the usability of application domains. To evaluate applications that have specific domain features, researchers can use sets of specific usability heuristics in addition to the well-known (usually Nielsen's) heuristics. Heuristics can also focus on the User eXperience (UX) aspects other than the usability. In a previous work, we proposed a formal methodology for establishing usability/UX heuristics. The methodology has 8 stages including activities to formulate, specify, validate and refine a new set of heuristics for a specific application domain. The methodology was validated through expert opinion and several case studies. Although when specifying the methodology, we explained each of its stages in detail, some activities can be difficult to perform without a guide that helps the researcher determine how the stages should be carried out. This article presents a detailed explanation regarding how to apply each stage of the methodology to create a new set of heuristics for a specific domain. Additionally, this paper explains how to iterate the methodology's stages and when to stop the process of developing new heuristics.

1. Introduction

Developing new heuristics as evaluation instruments has become a crucial activity to evaluate not only usability and User eXperience (UX) but also the specific features of different types of application domains. It is important to evaluate the application domain while considering all its components, features and the usability/UX attributes (or factors) that are critical in a specific application domain.

A widely used inspection method to evaluate usability, and the UX is a type of heuristic evaluation [1]. In this method, experts inspect a product/system/service to identify usability/UX problems and “measure” the usability/UX degree according to usability/UX principles or usability/UX heuristics [2]. Heuristics are rules or “guidelines” used to perform a heuristic evaluation [3]. Heuristic evaluation is a useful method to evaluate the usability/UX and detect potential problems. However, it may be necessary to use specific heuristics for the evaluation. Because each application domain may have unique and specific features, specific heuristics are recommended including elements that evaluate both usability/UX attributes/factors and their specific features [3].

There are methodologies that support the process of developing heuristics [4–10]; however, there has been no clear protocol for heuristic validation. We formalized the entire process of developing heuristics and proposed a formal methodology for establishing usability/user experience heuristics [11], including detailed activities to

formulate, specify, validate and refine a new set of heuristics. The methodology has 8 stages to develop heuristics, which considers the features of a specific application domain (product/system), the usability and the UX. The methodology was validated in several iterations [11]. Based on the results obtained, we concluded that the methodology is perceived as useful and that researchers will use it in the future [11]. In addition, we also concluded that the methodology is effective since it has been used to create new sets of heuristics (a final “product” is obtained) [11].

Although when specifying the methodology, we explained in detail each of its stages, some activities can be difficult to perform without a guide that helps the researcher determine how these stages should be developed. This article presents a detailed explanation regarding how to apply each stage of the methodology to create a new set of usability/UX heuristics for a specific application domain. The article also explains how to iterate the methodology's stages and when to stop the process of developing new heuristics. We aim to help the researcher during the heuristic development process using this methodology. To explain the process of developing heuristics, the development of a set of UX heuristics for national park websites is used as a case study [12]. In addition, several other case studies are analyzed to complement the explanation.

This document is organized as follows: Section 2 explores the theoretical background; Section 3 briefly presents the methodology for developing usability/UX heuristics; Section 4 explains in detail how to apply each methodology stage and how to iterate to create a new set of

* Corresponding author.

E-mail addresses: daniela.quinones@pucv.cl (D. Quiñones), cristian.rusu@pucv.cl (C. Rusu).

heuristics; and [Section 5](#) presents the conclusions and future studies.

2. Theoretical background

Usability is usually defined as the “capability of being used”. The complex and evolving nature of usability is difficult to describe in a unique definition. Usability refers to ease of use and the way users can perform their tasks. UX is a broader concept that includes usability. UX covers all aspects of an individual's interaction with a product, system and/or service. There are more than 86 methods to evaluate usability and UX [\[13\]](#). In general, evaluation methods can be classified into inspections, i.e., where expert evaluators participate; and tests, where users participate. One of the most widely used inspection methods is heuristic evaluation [\[1\]](#). The concepts of usability, heuristic evaluation, and user experience are briefly presented below.

2.1. Usability

Usability is more than “ease of use”. Usability involves a user interacting with a product, system or service; the goals that he/she expects to fulfill through the interaction; and the context of use, in addition to the efficiency, effectiveness and satisfaction associated with the interaction. The ISO 9241-11 standard defines usability as “the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use” [\[14\]](#). Usability is a complex issue [\[15\]](#). Each user (or group of users) may have different goals for the same product, system or service, depending on their needs. In addition, the interaction of each user can be different depending on the context of use. However, independent of the user, its goals and the context of use, the product, system or service must work in an efficient and effective manner, generating positive satisfaction for the user.

2.2. User experience

(UX) is a broader concept that includes usability; it involves not only the user's satisfaction, i.e., how pleasant it is to use a product, system or service, but also their emotions and perceptions during the interaction. The ISO 9241-210 standard defines the UX as “person's perceptions and responses resulting from the use and/or anticipated use of a product, system or service” [\[14\]](#). In other words, the UX includes all the beliefs, preferences, behaviors, physical and psychological responses that occur before, during and after use [\[14\]](#). Currently, it is not enough to design products that are useful and easy to use. Users also look for products that create pleasant experiences, generating a bond beyond the functional.

Several authors have proposed aspects, attributes or factors that define the UX. These factors encompass the different dimensions and/or features that are part of UX. Morville proposes seven UX facets or factors to illustrate the UX: useful, usable, desirable, findable, accessible, credible, and valuable [\[16\]](#). The facets are presented as a honeycomb diagram to help people understand the need to define priorities. Is it more important for a website to be desirable or accessible? How regarding usable or credible? The answer depends on the balance among the context of use, content and users [\[16\]](#).

Revang has suggested several factors that influence the UX [\[17\]](#). The author states that the user experience is a series of phases and that it is important to focus on positivity in the six main factors: usability, usefulness, findability, accessibility, credibility, and desirability. In turn, each factor is subdivided into other factors with a total of 30 factors. Arhipainen and Tähti have created a model that shows five elements that influence the UX [\[18\]](#). All these factors influence the experience that user product interaction evokes: social factors (such as time pressure), cultural factors (including language and habits), product factors (such as functions, usability, mobility, among others), user factors (e.g., emotions, skills, prior experiences, motivation, among

others), and context of use (such as time and place). Garret has proposed several elements of the UX divided into five planes [\[19\]](#): surface, skeleton, structure, scope, and strategy. These five planes provide a conceptual framework for discussing UX problems and the tools that one can use to solve them. These planes are not attributes but are presented as a guide for UX design.

It is important to consider each of the attributes/factors/facets of the UX when designing new products, systems or services. Likewise, to evaluate the UX, it is important to apply the most appropriate method(s) to evaluate all the factors or facets. Evaluating the UX is a challenge. A single evaluation method may not be sufficient to evaluate it completely. It is advisable to use both inspection methods and user tests to evaluate each UX aspect/attribute/factor. In this sense, the heuristic evaluation method allows evaluating the UX using a specific set of heuristics. However, since certain attributes of the UX are subjective (such as “desirable” and “valuable” [\[16\]](#)), and it can be difficult to fully evaluate the UX through a set of heuristics. A heuristic evaluation allows us to effectively evaluate certain attributes of the UX (useful, usable, findable, accessible, and credible [\[16\]](#)), but it is important to complement the evaluation using other methods to evaluate the UX completely.

To perform an effective evaluation of the UX and its factors, it is essential to use a set of specific heuristics. For this, it is possible to create a new set of heuristics if there is not one that evaluates the UX facets. Based on the results that have been presented in systematic literature reviews [\[20,21\]](#), apparently, there are no specific sets of heuristics to evaluate the UX and/or to evaluate certain attributes of the UX. However, it is critical to have sets of specific heuristics to correctly evaluate the UX through heuristic evaluation.

2.3. Heuristic evaluation

Heuristic evaluation is an inspection method used to evaluate the usability of a product interface. In addition, it is possible to evaluate certain UX aspects or attributes depending on the set of heuristics used. In this method, a set of expert evaluators inspect a product (interface) based on heuristics to identify usability/UX problems [\[2\]](#). The problems are classified (associated with the heuristics used) and rated (in terms of severity, frequency and criticality).

Heuristics are “rules of thumb” or “usability guidelines” used in a heuristic evaluation as an instrument for detecting usability problems [\[3\]](#). Nielsen's 10 heuristics [\[3\]](#) is one of the most commonly used sets of heuristics to evaluate systems or applications in general. However, since Nielsen's heuristics evaluate only general aspects or features of the systems, several sets of new usability heuristics have been developed to evaluate specific features of specific application domains. Based on the results presented in [\[20,21\]](#), more than 80 new sets of usability heuristics have been developed for specific domains.

3. The methodology

We proposed a formal methodology for developing usability/user experience heuristics for specific application domains [\[11\]](#). The methodology has 8 stages that can be iteratively applied to refine and improve the new set of proposed heuristics. We validated the methodology in two iterations through expert opinions. The validation results are detailed in [\[11\]](#). The methodology has already been applied to develop several new sets of heuristics for specific applications domains, such as online travel agency applications, websites, social networks, and videogames. All of these case studies proved the effectiveness of the methodology.

The methodology can be used to create heuristics and/or checklists that evaluate the usability or other UX aspects, such as playability, learnability, and communicability. This approach can also be used to create heuristic instruments for evaluating quality attributes other than usability (such as security and adaptability). [Fig. 1](#) shows the 8

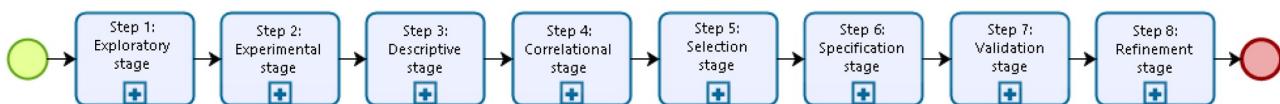


Fig. 1. Stages of the methodology to develop usability/user experience heuristics.

methodology stages. Although Fig. 1 sequentially shows the stages of the methodology, the development of heuristics may be performed iteratively. In specific situations, some stages may be optional, some stages overlap, and/or a stage may stop, and one can return to an earlier stage.

For each stage of the methodology, we specified (1) a definition, (2) the inputs that are needed to start the stage, (3) the activities that are performed in the stage, (4) the outputs that are obtained at the end of the stage, and (5) a BPMN diagram that graphically shows the activities of the stage [11]. The methodology is resumed in Table 1.

Although when specifying the methodology, we explained in detail each of its stages, some activities can be difficult to perform without a guide that helps the researcher determine how the stages should be developed. On the one hand, the methodology has several stages. While all stages are important for the development of a new set of heuristics, the researcher may be confused regarding how to apply them and what activities to carry out in each one (how to classify the information collected, what information to select, how to document the activities carried out, etc.). On the other hand, the methodology has a formalism that can be difficult to understand for the first time. The use of inputs,

Table 1
Definition, inputs and outputs for each stage of the methodology [11].

Name	Definition	Input	Output
Step 1: Exploratory stage	Perform a literature review	A specific application domain that needs a new set of heuristics or checklist	<ul style="list-style-type: none"> ① Information regarding the application (definitions and features) ② Usability and UX attributes ③ Sets of heuristics and/or other relevant elements ④ Additional specific features of the application
Step 2: Experimental stage	Analyze data that are obtained in different experiments to collect additional information that has not been identified in the previous stage	A specific application domain that needs a new set of heuristics or checklist	<ul style="list-style-type: none"> ⑤ Detected usability problems ⑥ Problems with existing heuristics ⑦ Selected information regarding the application ⑧ Selected features of the specific application domain ⑨ Selected usability/UX attributes ⑩ Selected sets of heuristics and/or relevant elements
Step 3: Descriptive stage	Select and prioritize the most important topics of all information that was collected in the previous stages	<ul style="list-style-type: none"> ⑪ Information regarding the application (definitions and features) ⑫ Usability and UX attributes ⑬ Sets of heuristics and/or other relevant elements ⑭ Additional specific features of the application ⑮ Detected usability problems ⑯ Selected information regarding the application ⑰ Selected features of the specific application domain ⑱ Selected usability/UX attributes ⑲ Selected sets of heuristics and/or other relevant elements 	<ul style="list-style-type: none"> ⑩ Selected sets of heuristics and/or relevant elements ⑪ Matched features, attributes and existing heuristics (and/or other relevant elements) ⑫ Categories
Step 4: Correlational stage	Match the features of the specific application domain with the usability/ UX attributes and existing heuristics (and/or other relevant elements)	<ul style="list-style-type: none"> ⑳ Selected features of the specific application domain ㉑ Selected usability/UX attributes ㉒ Selected sets of heuristics and/or other relevant elements ㉓ Selected sets of heuristics and/or other relevant elements ㉔ Matched features, attributes and existing heuristics (and/or other relevant elements) ㉕ Problems with existing heuristics ㉖ Matched features, attributes and existing heuristics (and/or other relevant elements) ㉗ Categories ㉘ Classified heuristics (to keep, adapt, create and eliminate) 	<ul style="list-style-type: none"> ㉙ Classified heuristics (to keep, adapt, create and eliminate) ㉚ Categories
Step 5: Selection stage	Keep, adapt and/or discard the existing sets of usability/UX heuristics that were selected in Step 3 (and/or other relevant elements)	<ul style="list-style-type: none"> ㉛ Selected sets of heuristics and/or other relevant elements ㉜ Matched features, attributes and existing heuristics (and/or other relevant elements) ㉝ Problems with existing heuristics ㉞ Matched features, attributes and existing heuristics (and/or other relevant elements) ㉟ Categories ㉟ Classified heuristics (to keep, adapt, create and eliminate) ㉟ Set of proposed heuristics 	<ul style="list-style-type: none"> ㉛ Classified heuristics (to keep, adapt, create and eliminate) ㉚ Set of proposed heuristics
Step 6: Specification stage	Formally specify the new set of usability/UX heuristics	<ul style="list-style-type: none"> ㉟ Set of proposed heuristics 	<ul style="list-style-type: none"> ㉚ Set of proposed heuristics
Step 7: Validation stage	Validate the set of heuristics through several experiments in terms of their effectiveness and efficiency in evaluating the specific application	<ul style="list-style-type: none"> ㉟ Set of proposed heuristics ㉟ Heuristic evaluation results: effectiveness of heuristics 	<ul style="list-style-type: none"> ㉟ Heuristic evaluation results: effectiveness of heuristics ㉟ Expert judgment results (survey) ㉟ User tests results: users' perceptions
Step 8: Refinement stage	Refine and improve the new set of heuristics based on the feedback that was obtained in Step 7	<ul style="list-style-type: none"> ㉟ Heuristic evaluation results: effectiveness of heuristics ㉟ Expert judgment results (survey) ㉟ User tests results: users' perceptions 	<ul style="list-style-type: none"> ㉟ Heuristic evaluation results: effectiveness of heuristics ㉟ Expert judgment results (survey) ㉟ User tests results: users' perceptions ㉟ Refining document: (1) What heuristics to create, refine and/or delete, why, and how to do it; (2) What steps to repeat

outputs, tables and BPMN diagrams are a great help in facilitating the heuristic development process, but at the beginning, it can be complex for the researcher to learn how to use them properly. For instance, in some stages, certain inputs and/or outputs are optional depending on the activities that are carried out. These options could confuse the researcher on how to start or finish the stages.

In addition, it is possible to repeat some stages of the methodology to refine and improve the set of proposed heuristics. For the researcher, it may not be evident which stages to repeat, at which time, or when to stop iterating. Due to the above, we believe that it is necessary to present a detailed explanation of how to apply each stage of the methodology. We aim to help the researcher in the heuristic development process using the methodology.

4. How to use the methodology in practice: a user guide

In this section, we explain in detail how to apply each methodology stage and how to iterate to create a new set of usability/UX heuristics for a specific application domain. The researcher can adapt the activities of the stages that he/she considers appropriate. We hope that this detailed explanation will serve as a guide for the researcher to apply the methodology to develop a new set of heuristics. This section is divided into two parts:

- 1 In [Section 4.1](#), we explain how to apply each stage. For each stage, we show the inputs, activities performed, outputs, and how to document the process. In addition, justifications are presented for each decision made regarding which activities to perform, which inputs/outputs to use, which information to select, etc.
- 2 In [Section 4.2](#), we explain how to iterate, when to iterate and how many iterations should be carried out to refine and improve the new set of heuristics. We include diagrams to show how to document the iterations.

To explain the process of developing heuristics, the development of a set of UX heuristics for national park websites is used as a case study [\[12\]](#). This case study is used to explain how to apply the methodology to develop heuristics that evaluate aspects other than usability, such as the UX. In addition, several other case studies are analyzed to complement the explanation. We include examples of other sets of heuristics for e-commerce, grid computing, and smartphones, among others.

4.1. How to use methodology's stages

The explanation of how to apply each stage is presented below.

4.1.1. Step 1: exploratory stage

• Input: specific application domain that needs a new set of heuristics or checklist. First, the researcher identifies a domain, for instance national park websites, smartphones, and E-commerce, that needs a new set of heuristics (or checklist) since there are no sets that effectively evaluate the usability/UX and their attributes. Then, the researcher must gather useful information for the development of heuristics, obtaining three outputs.

• Output: ② information regarding the application (definitions and features). Different information sources can be reviewed to collect information related to the specific domain. For instance, for national park websites, the researchers collected information regarding (1) the definitions of national parks and national park websites; (2) the purpose of a national park website; (3) the context of use of a national park website; (4) the advantages and disadvantages of using a national park website; and (5) the general and specific features of a national park website. General features are crosswise features for any kind of software product (and for that reason, they are also related to the national park websites). Specific features are features of the specific domain that differentiate them from other types of software products.

The researcher can gather other types of information that he/she considers useful for the development of heuristics, such as:

- 1 Types or classifications: A domain can be very broad and therefore have different classifications. For instance, for grid computing applications [\[22\]](#), the authors identified four types of grids: computer, data, service, and equipment grids.
- 2 Applications: Domain application areas. For instance, for grid computing applications [\[22\]](#), the authors identified four application areas: job submission, monitoring, visualization, and web portals.
- 3 Domain-specific tasks: Tasks that the user performs using products, systems or services of the specific domain. For instance, in [\[22\]](#), the authors have identified seven specific tasks that users perform using grid computing.
- 4 Target audiences: Users and/or types of users to whom the product, system or service of the specific domain is oriented.

• Output: ② usability and UX attributes. The researcher collects several attributes that define the usability/UX. We recommend reviewing different proposals stated by different authors to cover all the usability/UX dimensions, and then (in step 3) decide which of them is the most appropriate to evaluate the specific domain. For instance, to develop heuristics for national park websites, the researchers collected three proposals (two proposals for the usability and one for the UX): (1) usability attributes defined by the ISO 9241 standard [\[14\]](#); (2) usability attributes proposed by Nielsen [\[23\]](#); and (3) UX factors proposed by Morville [\[16\]](#). The researcher can gather other proposals regarding usability/UX attributes, such as (1) factors that influence the UX [\[17\]](#); (2) elements that influence the UX [\[18\]](#); (3) elements of UX [\[19\]](#); and (4) UX attributes [\[24\]](#).

• Output: ③ sets of heuristics and/or other relevant elements. The researcher collects an existing set of heuristics and/or other useful information (such as principles, design recommendations, guidelines, and patterns, related to the specific domain). The latter can be particularly useful when there are no other sets of related heuristics in addition to Nielsen's heuristics [\[3\]](#).

For example, to develop heuristics for national park websites, the researchers collected two existing sets of heuristics: (1) Nielsen's 10 heuristics (traditional heuristics) [\[3\]](#) and (2) heuristics for virtual museums (specific heuristics) [\[25\]](#). The researchers did not gather other relevant elements because they considered that with the information collected and the sets of heuristics reviewed, their collection was sufficient. In [\[26\]](#), for the u-learning domain, the authors selected two existing sets of heuristics to develop the new set: (1) Nielsen's 10 heuristics [\[3\]](#) and (2) heuristics for u-learning [\[27\]](#).

If the researcher is developing a set of heuristics for a specific application domain that is recent and/or emerging and for which there are no sets of specific heuristics that can serve as a basis, then we recommend reviewing other information that can provide important elements for creating the new heuristics. For example, in [\[28\]](#), the authors proposed a set of cultural-oriented usability heuristics. Since there were no sets of specific heuristics related to cultural aspects, the authors collected information on cultural dimensions and used those dimensions to define the new set of heuristics.

4.1.2. Step 2: experimental stage

• Input: specific application domain that needs a new set of heuristics or checklist. In this stage, the researcher can analyze data obtained in previous studies or perform their own experiments to collect additional information to develop the new set of heuristics. This step is optional. For instance, to develop heuristics for national park websites, the researchers did not perform this stage because (1) they performed an exhaustive literature review in the previous stage; (2) they found no previous studies related to national park websites' usability/UX; and (3) they did not have resources to perform the experiments.

The researcher decides if he/she needs to perform experiments.

Table 2

Information regarding national park websites prioritized.

Value	Information type	Description
3	Definition of the national park (physical place)	Definition proposed by the National Forestry Corporation (CONAF, Chile)
3	Definition of the national park (physical place)	Definition proposed by U.S. government
2	Definition of the national park (physical place)	Definition proposed by CITMA (website)
2	Purpose and definition of the national park (physical place)	Definition proposed by Real Academia Española [34]

Based on the previous data, the experiments performed (e.g., heuristic evaluation) and the way of interpreting the results, the researcher can obtain three different outputs. Since this stage is optional, the outputs are also optional. Depending on the activities performed, the researcher may have one to three outputs.

- **Output:** ④ *additional specific features of the application*. The researcher can analyze data from previous studies or perform new experiments to collect new features not identified in the previous stage. Although this output is optional, it is useful for Step 3: the Descriptive stage. For instance, to develop heuristics for grid computing applications [22], authors analyzed the results obtained by other authors regarding a specific application of grid computing [29] to identify additional specific features. In [28], for e-commerce websites, authors conducted interviews with expert evaluators to identify which elements of e-commerce should be evaluated with a set of heuristics.

- **Output:** ⑤ *detected usability problems*. The researcher can analyze data from previous studies or perform new experiments to identify usability problems related to the domain and create heuristics that evaluate and cover those problems. Although this output is optional, it is useful for Step 3: Descriptive stage.

For instance, to develop heuristics for smartphones [30] and cultural aspects [28], the authors performed a guided inspection to identify usability issues based on Nielsen's heuristics [3]. In [30], the authors carefully analyzed the issues that could not be associated with Nielsen's heuristics to identify which heuristics were necessary to create. In [31], for driving simulator applications, the researchers performed a heuristic evaluation to evaluate "SimuDrive" software using Nielsen's heuristics to identify usability issues and then determine which problems are not possible to associate with Nielsen's heuristics.

- **Output:** ⑥ *problems with existing heuristics*. The researcher can analyze data from previous studies or perform new experiments to identify which problems occur while understanding existing heuristics and to avoid them. In addition, the researcher can find interesting elements to include in the new set of heuristics by reviewing existing sets. Although this output is optional, it is useful for Step 6: Specification stage.

For instance, to develop heuristics for transactional websites [32,33], the authors analyzed the results obtained in 6 heuristic evaluations performed by other evaluators to determine the problems using Nielsen's heuristics when evaluating the website www.hotelclub.com. This analysis allowed the identification of the present heuristics problems that need to be understood.

4.1.3. Step 3: descriptive stage

- **Inputs:** ① *information regarding the application (definitions and features)*; ② *usability and UX attributes*; ③ *sets of heuristics and/or other relevant elements*; ④ *additional features of the application*; and ⑤ *detected usability problems*. For step 3, inputs ①, ②, and ③ are mandatory. Inputs ④ and ⑤ are optional, since they depend on whether the researcher performed step 2 or not. For instance, to develop heuristics for national park websites, the researchers did not work with inputs ④ and ⑤ since step 2 was not performed.

In the descriptive stage, the researcher selects and prioritizes the information collected in steps 1 and 2 (if it applies). The information is selected and grouped into five topics: (1) information regarding the specific domain; (2) features of the specific domain; (3) usability/UX

attributes; (4) existing sets of heuristics and/or other relevant elements; and (5) usability problems detected. The last topic (5) is optional, depending on whether the researchers performed step 2. Then, the information is prioritized by each topic using a scale of 3 levels (3: highly important; 2: somewhat important; and 1: not important). As the information is prioritized, the outputs of step 3 are generated.

To develop heuristics for the national park websites, the researchers selected and grouped the information in the corresponding topics (except for topic 5, since they did not perform step 2) [11]. However, they prioritized the information according to what they considered important, without using the scale proposed in the methodology due to time limitations. To explain how to use the scale, we present below how the researchers should have applied the scale for the development of heuristics for national park websites.

- **Output:** ⑦ *selected information regarding the application*. The researcher orders and prioritizes the information collected regarding the domain (definitions, classifications, areas of use, etc.) using input ① (information regarding the application).

To develop heuristics for national park websites, the researchers prioritized the information, as shown in Table 2. This table shows the type of information collected, a brief description, and the value assigned according to their importance.

Since there was no definition for national park websites, the researchers proposed one. After reviewing and analyzing their work [11], we identified that they proposed the definition using the information qualified with value 3 (see Table 2).

- **Output:** ⑧ *selected features of the specific application domain*. The researcher orders and prioritizes the information collected regarding the features of the specific domain using inputs ① (information regarding application) and ④ (additional features of the application, if it applies). The researcher can group the features into general (crosswise features for any kind of software product) and specific features (features of the specific domain that differentiate them from other types of software products).

To develop heuristics for the national park websites, the researchers identified 8 general features and 9 specific features [11]. After reviewing and analyzing their work [12], we detected the following:

- 1 The researchers selected all general features collected in step 1, which means that they had to rate each feature with value 3 (the highest, indicating that all general features are important to consider for the development of heuristics).
- 2 The researchers selected 8 of 9 specific features collected in step 1, which means that they had to rate 8 specific features with value 3 (the highest) and 1 specific feature with value 1 (the lowest, indicating that this feature is not important, e.g., because it is not related to the domain).

- **Output:** ⑨ *selected usability/UX attributes*. The researcher orders and prioritizes the information collected regarding usability/UX attributes using input ② (usability and UX attributes). The researcher can prioritize the usability and UX attributes separately or together. Additionally, the researcher can prioritize the attributes by following two approaches (or the mixture of both):

- 1 Make a list of all the identified attributes, regardless of the author

Table 3
Prioritizing sets of attributes.

	Value	Author of the proposal	Amount of attributes	Proposal of attributes	Justification
Usability	3	Nielsen [23]	5	Learnability, efficiency, memorability, errors and satisfaction.	Since Nielsen proposal includes more usability attributes than the ISO standard, a value of priority 3 and 2 has been assigned.
UX	2	ISO 9214 [14]	3	Effectiveness, efficiency and satisfaction.	
	3	Morville [16]	7	Useful, usable, desirable, findable, credible, accessible and valuable.	Only the UX factors proposed by Morville were reviewed.

that proposed them.

2 Make a list of a set of attributes, proposed by different authors.

That is, the researcher decides if he/she wants to prioritize considering all the attributes or if he/she wants to prioritize the proposals of attributes suggested by a particular author. Regardless of the approach chosen to prioritize, the researcher must justify why he/she has chosen certain attributes and why he/she has discarded others.

For instance, to develop heuristics for national park websites, the researchers collected three proposals of attributes for the usability/UX (two proposals for usability and one for UX, see step 1). After reviewing and analyzing their work [12], we detected that the researchers performed a prioritization at two levels. First, they prioritized the set of attributes proposed by different authors for usability and UX (see Table 3), and then they selected which attributes of each proposal will be used for the development of the heuristics (see Table 4). Tables 3 and 4 show how the researchers prioritized the information using the scale proposed in the methodology (column named “value”). In addition, these tables also show the justification of why certain proposal/attributes have been selected and/or discarded (column named “justification”).

- **Output: ② selected sets of heuristics and/or other relevant elements.** The researcher orders and prioritizes the information collected regarding the sets of heuristics and/or other relevant elements using inputs ③ (sets of heuristics and/or other relevant elements) and ⑤ (detected usability problems, if it applies).

The researcher can select one or more sets of existing heuristics. On the one hand, if there are no specific sets to evaluate similar domains, then we recommend using Nielsen's heuristics [3] as a basis. On the other hand, if there are other similar sets, then the researcher can choose those that he/she considers useful (the scale to prioritize can help to define which set is the most appropriate). In addition, if there are no similar sets of heuristics, then other relevant elements collected can serve as a basis to develop the new heuristics (patterns, guidelines, etc.). For example, usability problems detected in step 2 can help to identify which heuristics need to be created to detect usability problems related to the specific domain. The researcher makes a list of the detected usability problems and determines which elements of the domain should be evaluated with the new set of heuristics.

For instance, to develop heuristics for national park websites, the researchers identified two sets of heuristics: Nielsen's heuristics (traditional heuristics) [3] and heuristics for virtual museums (specific heuristics) [25]. No other relevant elements were collected. The researchers selected heuristics for virtual museums since Nielsen's heuristics are too generic and do not cover some specific features of national park websites. This means that the researchers had to rate heuristics for virtual museums with value 3 (the highest) and Nielsen's heuristics with value 1 (the lowest).

In [28], to develop cultural-oriented usability heuristics, the authors identified three sets of heuristics (1 set of traditional heuristics and 2 sets of specific heuristics) and other relevant elements (cultural dimensions and cultural aspects in websites, models and guidelines). Although authors in [28] did not prioritize the information collected using the scale proposed in our methodology, they prioritized the importance of the information. Table 5 shows what the values assigned to each element would be based on the selection made by the authors in [28] and its justification.

Depending on the specific domain, a researcher can find several sets of existing heuristics that may be used for the development of new heuristics. If the researcher finds a large number of sets of existing heuristics, then it can be difficult for him/her to prioritize the sets. To facilitate the prioritization, we recommend performing a comparative analysis between sets of heuristics first and then prioritizing them. The researcher can review the heuristics of each set and match those that evaluate the same aspects. Table 6 shows an example of how to analyze different sets of heuristics for the e-commerce website domain. As

Table 4
Prioritizing attributes individually.

	Value	Attribute	Author of the proposal	Justification
Usability	3	Learnability	Nielsen [23]	The 5 attributes proposed by Nielsen were selected since they are more complete than those proposed by the ISO standard.
	3	Efficiency		
	3	Memorability		
	3	Errors		
	3	Satisfaction		
	3	Useful	Morville [16]	Six of the seven factors proposed by Morville were selected. “Accessible” was not considered, as it is not intended to evaluate the accessibility with the new set of heuristics.
	3	Usable		
	3	Desirable		
	3	Findable		
	3	Credible		
	1	Accessible		
	3	Valuable		

shown in [Table 6](#), based on the comparative analysis, it is possible to identify aspects that are evaluated by a single set of heuristics or by several of them. In addition, this analysis allows the determination of whether the existing sets of heuristics evaluate all the features of the specific domain in study. It is even possible that after reviewing the existing sets of heuristics, the researcher determines that a certain set properly evaluates the specific application domain, and therefore, he/she may stop the process of developing new heuristics.

4.1.4. Step 4: correlational stage

- **Inputs:** ① selected information regarding the application; ② selected features of the specific application domain; ③ selected usability/UX attributes; and ④ selected sets of the heuristics and/or other relevant elements. In this stage, the researcher performs two activities: (1) Match the features of the specific application domain, the usability/UX attributes, and the existing heuristics (and/or other relevant elements); and (2) Group heuristics into categories (if it applies).

- **Output:** ⑤ matched features, attributes and existing heuristics (and/or other relevant elements). The researcher performs the feature match using all the inputs (⑦, ⑧, ⑨, and ⑩). To perform the match, the researcher first creates a list with all the features of the specific domain. Then, he/she matches each feature with the usability/UX attribute. Finally, the researcher analyzes whether the existing heuristics selected completely or partially cover each of the features and usability/UX attributes. It is normal that some features are not covered by an existing heuristic (this means that it will be necessary to create a heuristic to evaluate those features).

For instance, to develop heuristics for national park websites, the researchers matched the general and specific features of national park websites, the usability/UX attributes, and the set of heuristics for virtual museums [\[11\]](#). Researchers considered both the usability and UX attributes, and for this reason, each feature has associated usability and/or UX attributes. In addition, some existing heuristics partially cover some features and attributes, while other features and attributes are not covered by any existing heuristic.

In [\[28\]](#), to develop cultural-oriented usability heuristics, the authors performed a match among cultural dimensions (other relevant element), features and the heuristics created. Authors did not match the usability attributes, but they did use the usability attributes proposed by Nielsen to create the new heuristics [\[23\]](#). Based on the information reviewed in [\[28\]](#), it was possible to determine how the authors produced the matching. [Table 7](#) shows the match among features, usability attributes, Nielsen's heuristics and cultural dimensions. As shown in [Table 7](#), it is possible to make a match with other relevant elements (in this case, cultural dimensions) in addition to the existing heuristics.

- **Output:** ⑥ categories. The researcher decides whether the new

heuristics will be grouped into categories or will not be based on the features identified (it is not mandatory to define categories).

Depending on the specific application domain, it may be useful to group heuristics into categories. Each category can group heuristics that evaluate certain specific aspects. To create and/or name the categories, the output ⑪ obtained in this same stage can be useful.

For instance, to develop heuristics for national park websites, the researchers did not define categories since each heuristic evaluates a different feature of the specific domain. In [\[22\]](#), to develop heuristics for grid computing applications, authors classified the heuristics into three categories: (1) Design and aesthetics heuristics; (2) Flexibility and navigation heuristics; and (3) Errors and help heuristics. To develop playability heuristics for mobile games [\[41\]](#), the authors grouped the heuristics into three categories: (1) Game usability heuristics; (2) Mobility heuristics; and (3) Gameplay heuristics. In [\[42\]](#), the authors created four categories to group the heuristics for educational games: (1) Interface heuristics; (2) Content heuristics; (3) Pedagogical/educational heuristics; and (4) Multimedia heuristics.

4.1.5. Step 5: selection stage

- **Inputs:** ⑦ selected sets of heuristics and/or other relevant elements; and ⑧ matched features, attributes and existing heuristics (and/or other relevant elements). In the selection stage, the researcher reviews existing heuristics and determines which heuristics to keep, adapt, create and/or eliminate to develop the new set of heuristics. For the above, the researcher uses input ⑨ to review each existing heuristic and input ⑩ to identify which new heuristics are necessary to create.

- **Output:** ⑨ classified heuristics (to keep, adapt, create and eliminate). The researcher classifies the heuristics using all the inputs. Using input ⑦ (selected sets of heuristics and/or other relevant elements), the researcher reviews each existing heuristic and decides which heuristics to keep, adapt, or eliminate. When reviewing different sets of heuristics, it is possible to identify very similar heuristics obtained from different sets. In this case, it is necessary to eliminate the redundancy and complement the heuristic's information if necessary but without repeating it.

Using input ⑩ (matched features, attributes and existing heuristics, and/or other relevant elements), the researcher can identify which new heuristics are necessary to create to evaluate specific domain-related features of the specific domain. If the researcher analyzed other relevant elements (already included in output ⑪), then these can be helpful in determining which new heuristics to create.

For instance, the selection process performed to develop the heuristics for national park websites is synthesized in [\[11\]](#) and presented in detail in [\[12\]](#). The authors analyzed the existing set of 15 heuristics for virtual museums but did not analyze other relevant elements. Using

Table 5
Sets of heuristics and other relevant elements prioritized for cultural-oriented usability heuristics [28].

Type of information	Description	Details	Value	Justification
Sets of heuristics	Nielsen's heuristics [3]	10 usability heuristics	3	10 Nielsen's heuristics were selected since authors did not find specific cultural-oriented usability heuristics
	Quiñones et al. heuristics [32]	13 usability heuristics for transactional web applications	1	This set of heuristics was not selected since they do not evaluate cultural aspects
	Bonastre and Granollers heuristics [35]	64 usability heuristics for e-commerce websites (presented as questions)	1	This set of heuristics was not selected since the study did not present experimental validation
Other relevant elements	Hofstede's cultural dimensions [36]	<ul style="list-style-type: none"> 1. Power distance dimension (PD) 2. Individualism dimension (IDV) 3. Masculinity dimension (MAS) 4. Uncertainty avoidance dimension (UA) 5. Long-term orientation dimension 6. Indulgence and restraint dimension <p>Model for developing usable cross-cultural websites (Smith et al. [37])</p>	3	PD, IDV, MAS, and UA dimensions were selected since the authors considered that they help to differentiate cultures [28]
	Cultural aspects in websites	<p>Guidelines for cultural-oriented interfaces (cross-cultural effects in e-retailing, Huggins et al. [38])</p> <p>Guidelines for cultural-oriented interfaces (effects of usability and web design attributes on user preference for an e-commerce website, Sangwon et al. [39])</p>	3	The 3 proposals were selected since all of them provide relevant information regarding cultural aspects for different types of websites [28]

input ⑩, the researchers reviewed 15 heuristics, of which 0 heuristics were kept unchanged, 12 were adapted, and 3 heuristics were eliminated, as they were not applicable to the evaluation of national park websites. Using output ⑪, the authors determined that it was necessary to create 6 new heuristics to evaluate the specific features of the national park websites.

In [28], to develop cultural-oriented usability heuristics, the authors also performed a heuristic selection process, although they did not document it, as it is proposed in our methodology. Based on the information reviewed in [28], it was possible to determine how the authors would have performed the selection process. Table 8 shows some examples of the heuristic selection process for creating the set of heuristics using the existing set of Nielsen's heuristics [3]. The authors reviewed 10 heuristics, and all were adapted (none were kept or eliminated). Each heuristic was adapted to evaluate related cultural aspects, and its specification was detailed. In addition, the investigators used cultural dimensions to develop a new set of heuristics (as "other relevant elements"). Based on the above, these authors created 2 new heuristics to evaluate specific features.

4.1.6. Step 6: specification stage

• **Inputs:** ⑥ problems with existing heuristics; ⑪ matched features, attributes and existing heuristics (and/or other relevant elements); ⑫ Categories; and ⑬ classified heuristics (to keep, adapt, create and eliminate). In the specification stage, the researcher defines the amount of heuristics; determines the elements to include in the heuristics' specification; groups the heuristics into categories (if it applies); and formally specifies the new set of heuristics using a template. We recommend using a standard template because all the heuristics will be consistent in the information they present, and each heuristic will have a level of detail that helps to understand it better. The researcher may decide whether to use the complete template or just some elements to specify the heuristics.

We recommend keeping the number of heuristics relatively small (between 10 and 16) [20], and if it is necessary to add a greater level of detail, then we suggest developing an additional checklist. The standard template proposed to specify the heuristics can be reviewed in [11].

• **Output:** ⑭ set of proposed heuristics. To specify a new set of heuristics, inputs ⑪ and ⑬ are mandatory. Inputs ⑥ and ⑫ are optional since input ⑥ depends on whether the researcher performed step 2 or not, and input ⑫ depends on whether the researcher creates categories in step 4.

For instance, to develop heuristics for national park websites, the researchers did not work with inputs ⑥ and ⑫ because step 2 was not performed and no categories were defined to group the heuristics in step 4. Using inputs ⑪ and ⑬, the researchers created 14 heuristics and used the 12 elements of the template to specify the heuristics. The set of heuristics created can be reviewed in [11] (the complete version can be reviewed in [12]).

In [30], to develop heuristics for smartphones, the authors develop 12 heuristics; and in [22], the authors develop 10 heuristics for grid computing applications, grouping the heuristics into three categories. In both articles, the authors specify heuristics using the following 7 elements: ID, name, definition, explanation, examples, benefits and problems associated with misinterpretation. In both cases, the authors used the methodology proposed by Rusu et al. [43] to create the new set of heuristics. Our methodology is based on [43], so the template is similar. However, we have added more elements to the specification template.

4.1.7. Step 7: validation stage

• **Inputs:** ⑮ Set of proposed heuristics

In this stage, the researcher validates the new set of proposed heuristics through several experiments. These experiments may check

Table 6

Comparative analysis between four sets of heuristics to evaluate transactional websites.

Nielsen's heuristics [3]	Paz et al. heuristics [40]	Quiñones et al. heuristics [33]	Díaz et al. heuristics [28]
N1: Visibility of the system status	T1: Visibility and clarity of the system's elements T2: Visibility of the system status	SWT1: Visibility of the system status	CH1: Visibility of the system status
N2: Match between the system and the real world	T3: Match between the system and user's cultural aspects T4: Feedback of the transaction	SWT5: Match between the system and the real world SWT2: Keep the user informed regarding the status of transactions SWT8: Use of standards and web symbology	CH2: Match between the system and the real world –
N4: Consistency and standards	T5: Alignment to the web standards design T7: Standard iconography T6: Consistency of design	SWT7: Consistency between the elements of the system SWT4: Security and speed of the transactions	CH4: Consistency and standards –
–	–	–	CH11: The information structure
–	–	–	–

the effectiveness and efficiency of the new set of heuristics when evaluating the usability/UX of specific domains. For each experiment, input ⑩ is used. Based on the experiment performed, three outputs can be obtained:

- 1 Experiment: Heuristic evaluations. Output: ⑪ Heuristic evaluation results: effectiveness of heuristics.
- 2 Experiment: Expert judgments. Output: ⑫ Expert judgment results (survey): utility, clarity, ease of use, need for checklist and comments regarding each heuristic.
- 3 Experiment: User tests. Output: ⑬ User tests results: users' perceptions.

For instance, to develop heuristics for national park websites, the researchers validated the heuristics through heuristic evaluation and expert judgment (survey) in the first iteration and through user tests (co-discovery and focus group) in the second iteration.

In [26], to establish heuristics for u-learning applications, the authors validated the heuristics in two iterations through expert judgment (survey) and two heuristic evaluations. In [31], to create heuristics for driving simulators, authors validated the heuristics through expert opinion and three heuristic evaluations.

To develop cultural-oriented heuristics [28], the authors validated the heuristics through two heuristic evaluations in the second iteration, through three heuristic evaluations in the third iteration, and through expert judgment (survey) in the fourth iteration.

4.1.7.1. Validation through heuristic evaluation. The researcher checks the new set of heuristics against control heuristics (traditional or

specialized heuristics). For this, the researcher (1) selects applications to evaluate; (2) selects the set of control heuristics; (3) selects evaluators for a heuristic evaluation with similar experience (define control and experimental groups); and (4) evaluates the effectiveness of the heuristics in terms of the 5 criteria. The new set of usability/UX heuristics performs well, and it is an effective instrument when better results than the control heuristics are obtained in terms of the 5 criteria [11]. Table 9 shows how different authors validate the set of proposed heuristics through heuristic evaluations.

The evaluations involved between 2 and 5 evaluators per group. Some evaluations involved more than one control or experimental group, so column "Number of evaluators involved" shows more evaluators in certain cases. In addition, in some evaluations, the authors worked with only the experimental group (without the control group) because it was the last iteration (heuristics for smartphones [30], cultural-oriented heuristics [28]). The last column of the table shows six criteria used by different authors to validate the heuristics. Each criterion is detailed below.

- 1 C1: Numbers of correct and incorrect associations of problems to heuristics
- 2 C2: Number of usability/UX problems identified
- 3 C3: Number of specific usability/UX problems identified
- 4 C4: Number of identified usability/UX problems that qualify as more severe
- 5 C5: Number of identified usability/UX problems that qualify as more critical
- 6 C6: Average of Hofstede' cultural dimensions.

Table 7

Match among features, usability attributes, Nielsen's heuristics and cultural dimensions [28].

Feature	Usability attribute	Nielsen's heuristic [3]	Cultural dimension
Informed actions – status changes – feedback	Memorability	N1: Visibility of the system status	(UA) Uncertainty avoidance
Familiar concepts – conventions	Memorability	N2: Match between the system and the real world	(IDV vs COL) Individualism vs. collectivism
Control – undo/redo	Efficiency	N3: User control and freedom	(MAS vs FEM) Masculinity vs. femininity
Conventions – standards	Satisfaction	N4: Consistency and standards	(IDV vs COL) Individualism vs. collectivism
Avoid mistakes – detailed design – prevention	Errors	N5: Error prevention	(UA) Uncertainty avoidance
Memory load – remember information – instructions	Memorability	N6: Recognition rather than recall	(UA) Uncertainty avoidance
Efficient – simple – efficacy	Efficiency	N7: Flexibility and efficiency of use	(MAS vs FEM) Masculinity vs. femininity
Clean interface – significant elements	Satisfaction	N8: Aesthetic and minimalist design	(IDV vs. COL) Individualism vs. collectivism
Clear content – suggesting solution – details	Errors	N9: Help users recognize, diagnose, and recover from errors	(UA) Uncertainty avoidance
Documentation – information – adequate	Learnability	N10: Help and documentation	(UA) Uncertainty avoidance
Hierarchic – organization – options	Efficiency	–	(PD) Power distance
Goal oriented – concise	Learnability	–	(MAS vs FEM) Masculinity vs. femininity

Table 8

Heuristic selection process for creating cultural-oriented heuristics.

ID	Heuristic Name/Explanation	Action	Set of existing heuristics	Aspect or feature covered	Applicability
N3	User control and freedom	Adapt	Nielsen's heuristics [3]	Control – undo/redo (partially covered)	(3) Critical
N7	Flexibility and efficiency of use	Adapt	Nielsen's heuristics [3]	Efficient – simple –efficacy (partially covered)	(3) Critical
CH11	The information structure	Create	–	Hierarchic – organization – options	(3) Critical
CH12	Accurate and detailed results	Create	–	Goal oriented – concise	(3) Critical

Table 9

How different authors validate a new set of heuristics through heuristic evaluations.

Set of proposed heuristics	Control heuristics selected	Number of evaluators involved		Criteria used to evaluate					
		Control group	Experimental group	C1	C2	C3	C4	C5	C6
UX heuristics for national park websites [11]	Usability heuristics for virtual museums [25]	3	3	x	x	x	x	x	x
Usability heuristics for u-learning applications [26]	Nielsen's heuristics [3]	2	2	x	x	x	x	x	x
Usability heuristics for driving simulators [31]	Nielsen's heuristics [3]	3	3	x	x	x	x	x	x
Cultural-oriented usability heuristics [28]	Nielsen's heuristics [3]	3	3	x	x	x	x	x	x
Smartphone's usability heuristics [30]	–	0	21–27	x	x	x	x	x	x
–	Nielsen's heuristics [3]	2	2	x	x	x	x	x	x
–	Nielsen's heuristics [3]	3	3	x	x	x	x	x	x
–	–	0	27	x	x	x	x	x	x
Usability heuristics for grid computing applications [22]	Nielsen's heuristics [3]	4	4	x	x	x	x	x	x
Usability heuristics for transactional websites [33]	Nielsen's heuristics [3]	3	3	x	x	x	x	x	x
	Nielsen's heuristics [3]	19	4	x	x	x	x	x	x

The first 5 criteria have been proposed in our methodology [11]. The last criterion (C6) is specific for the area of application (cultural aspects [28]), so it only applies to that case study.

• Output: Heuristic evaluation results: effectiveness of heuristics

The researcher evaluates the effectiveness of the new set of heuristics based on 5 criteria. Each criterion is explained below.

1 Numbers of correct and incorrect associations of problems to heuristics

The new set of heuristics is an effective instrument when it has fewer incorrect associations of the identified usability/UX problems to the heuristics than that of the control heuristics.

The researcher must perform a critical and constructive analysis of the results that are obtained by the control group and experimental group. The researcher must (critically, without influencing the results) analyze the associations of the problems that are detected with the heuristics and determine whether each association is correct or incorrect.

This analysis provides qualitative data, in which it is possible to identify why the evaluators incorrectly associated the problems with certain heuristics and solved the problems (if any). The following problems are considered:

- 1 The definition of a heuristic is unclear. It is necessary to improve its definition.
- 2 There are heuristics with similar definitions. It is necessary to clearly differentiate the heuristics and the aspects that they evaluate.
- 3 There are heuristics that evaluate the same aspect (redundancy). It is necessary to eliminate a heuristic.
- 4 The evaluators did not understand how to use the heuristics. The heuristics are well specified, but the evaluators did not know how to use them correctly.
- 5 Some heuristics do not have associated problems. These heuristics should not be discarded but should be analyzed with respect to why they do not have associated problems.

To measure the effectiveness of the new set of heuristics compared to the control heuristics, four percentages must be calculated using the following Eq. (1):

$$CA = \frac{\sum_{n=1}^T CAH_n}{TP} \times 100 \quad (1)$$

where

- CA: correct associations;
- T: total number of heuristics of the set;
- CAH_n: number of correct associations of the problems to the heuristic "n";
- TP: total usability/UX problems identified.

Two percentages should be calculated:

- 1 Percentage of correct associations of problems to the new set of heuristics (CA1);
- 2 Percentage of correct associations of problems to the control heuristics (CA2).

The new heuristics perform well when CA1 is greater than or equal to CA2, that is, when the percentage of correct associations of the new set of heuristics is greater than or equal to the percentage of correct associations of the control heuristics.

The effectiveness in terms of the number of incorrect associations of the problems to the heuristics can be represented as a percentage by the following Eq. (2):

$$IA = \frac{\sum_{n=1}^T IAH_n}{TP} \times 100 \quad (2)$$

where:

- IA: incorrect associations;
- T: total number of heuristics of the set;
- IAH_n: number of incorrect associations of the problems to the heuristic "n";

- TP: total usability/UX problems identified.

Two percentages should be calculated:

- 1 Percentage of incorrect associations of the problems to the new set of heuristics (IA1);
- 2 Percentage of incorrect associations of the problems to the control heuristics (IA2).

The new heuristics perform well when IA1 is less than IA2, that is, when the percentage of incorrect associations of the new set of heuristics is less than the percentage of incorrect associations of the control heuristics. Formulas (1) and (2) are applicable when each problem is associated with only one heuristic. When a problem is associated with more than one heuristic, CA and IA are difficult to interpret.

2 Number of usability/UX problems identified

Three categories of problems are expected:

- 1 (P1) Problems that are identified by both groups of evaluators (common problems identified by both groups),
- 2 (P2) Problems that are identified only by the group that used the new set of heuristics (without considering the common problems),
- 3 (P3) Problems that are identified only by the group that used control heuristics (without considering the common problems).

The new heuristics perform well when (P1) and/or (P2) include the highest percentage of problems. A question arises when (P3) has the highest percentage of problems: Why are these problems not identified when using the new set of heuristics? There are two possible reasons:

- 1 The new heuristics are not able to identify these problems, either because there are no appropriate heuristics or because the heuristics are not properly specified.
- 2 Evaluators who used the new heuristics subjectively ignored the problems.

Suppositions 1 and 2 may be validated or rejected by expert judgment, complementary evaluations and/or user tests:

- 1 Through expert judgment, it is possible to determine whether heuristics are poorly defined and/or misunderstood or whether it is necessary to include a new heuristic to evaluate certain aspects of the application.
- 2 By performing a user test, it is possible to determine whether usability problems that are detected only by the group that used the control heuristics are or are not perceived as real problems by users.

To avoid the scenario in which evaluators subjectively ignore usability problems, the following steps are suggested:

Table 10
Rating scales of detected usability/UX problems.

Rating scale	Description	Values	Scale
Frequency	Frequency indicates how often the detected usability/UX problem occurs	0–4	4: > 90% 3: 51–90% 2: 11–50% 1: 1–10% 0: < 1%
Severity [44]	Severity indicates how catastrophic the detected usability/UX problem is	0–4	4: Catastrophic problem 3: Major problem 2: Minor problem 1: "Cosmetic" problem 0: Not a problem
Criticality	Sum of the frequency and severity values.	0–8	–

1 Check if the new set of heuristics covers all the attributes of usability/UX and application features that are defined in Step 4: Correlational stage.

2 Instruct (or remind) the evaluators to analyze together the detected usability/UX problems. Evaluators should review each usability/UX problem and determine if all aspects of usability/UX and application features have been evaluated. (It is not possible to completely eliminate subjectivity from the experiment, which is related to the evaluators' experience and the correct use of heuristics).

Explanations through case studies on how to apply this criterion can be reviewed in [22,28,30].

3 Number of specific usability/UX problems identified

The new set of heuristics is an effective instrument if more specific usability/UX problems – related to the evaluated application – are identified compared to the control heuristics (i.e., usability/UX issues that are related to specific features or specific aspects of the application).

The effectiveness in terms of the number of specific usability/UX problems that are identified (ESS) can be represented as a percentage by using the following Eq. (3):

$$ESS = \frac{NSP}{TP} \times 100 \quad (3)$$

where:

- ESS: effectiveness;
- NSP: number of specific usability/UX problems identified;
- TP: total usability/UX problems identified.

Two effectiveness values should be calculated:

- 1 The effectiveness using problems that were identified only by the group that used the new set of heuristics (ESS1).
- 2 The effectiveness using problems that were identified only by the group that used control heuristics (ESS2).

The new heuristics perform well when ESS1 is at least equal to (but hopefully greater than) ESS2, that is, when the new set of heuristics finds more specific usability/UX problems than the control heuristics.

4 Number of identified usability/UX problems that qualify as more severe

In addition, the new set of heuristics is an effective instrument if it finds more usability/UX problems that qualify as the most severe. In a heuristic evaluation, after preparing the unique and complete list of usability/UX problems, each evaluator separately qualifies the

Table 11

The effectiveness of the new set of heuristics for national park websites.

	Experimental group	Control group	Observations
Set of heuristics used	Heuristics for national park websites (NPH)	Heuristics for virtual museums (VMH)	-
Amount of heuristics (T)	18	15	-
Total of problems identified (TP)	46	47	-
Number of specific problems identified (NSP)	12	9	-
Number of problems identified and qualified with a severity greater than 2 (NPV)	28	21	-
Number of problems identified and qualified with a criticality greater than 4 (NPC)	27	24	-
Problems identified by both groups (P1)	7		Given that (P1) and/or (P2) do not include the highest amount of problems, it is necessary to perform additional experiments to review and improve NPH.
Problems identified by the experimental group (P2)	39	-	
Problems identified by the control group (P3)	-	40	
Total of the correct associations (ΣCAHn)	32	28	-
Total of the incorrect associations (ΣIAHn)	14	19	-
Percentage of the correct associations (CA)	CA1 = 69.6%	CA2 = 59.6%	Given that CA1 > CA2 it is concluded that NPH works better than VMH (NPH has a higher percentage of correct associations)
Percentage of the incorrect associations (IA)	IA1 = 30.4%	IA2 = 40.4%	Given that IA1 < IA2 it is concluded that NPH works better than VMH (NPH has a lower percentage of correct associations)
Effectiveness in terms of number of specific problems identified (ESS)	ESS1 = 26.1%	ESS2 = 19.1%	Given that ESS1 > ESS2 it is concluded that NPH works better than VMH (NPH finds more specific usability/UX problems than VMH)
Effectiveness in terms of number of problems identified and qualified with a severity greater than 2 (ESV)	ESV1 = 60.8%	ESV2 = 44.6%	Given that ESV1 > ESV2 it is concluded that NPH works better than VMH (NPH finds that more usability/UX problems qualify as more severe than VMH)
Effectiveness in terms of number of problems identified and qualified with a criticality greater than 4 (ESC)	ESC1 = 58.6%	ESC2 = 51%	Given that ESC1 > ESC2 it is concluded that NPH works better than VMH (NPH finds that more usability/UX problems qualify as more critical than VMH)

problems. We qualify the problems according to their frequency, severity, and criticality (see Table 10). In this case, the effectiveness depends on the severity since usability/UX problems with high severity impede the proper functioning of the application, thereby generating errors.

The rating scale for severity ranges from 0 to 4, where 0 indicates that the problem is not severe and 4 indicates that the problem is catastrophic. In this sense, it is important to identify those problems that have a severity value that is greater than 2, as this is the average value on the scale of severity. If the researcher decides to use another rating scale for severity, then the average value of that scale should be used in the formula that is presented below. The effectiveness in terms of the number of identified usability problems that qualify as more severe (ESV) can be represented as a percentage by using the following Eq. (4):

$$\text{ESV} = \frac{\text{NPV}}{\text{TP}} \times 100 \quad (4)$$

where:

- ESV: effectiveness;
- NPV: number of usability/UX problems identified qualified with a severity greater than 2;
- TP: total usability/UX problems identified.

Two effectiveness values should be calculated:

- 1 The effectiveness using problems that were identified only by the group that used the new set of heuristics (ESV1).
- 2 The effectiveness using problems that were identified only by the group that used control heuristics (ESV2).

The new heuristics perform well when ESV1 is at least equal to (but hopefully greater than) ESV2. That is, the new set of heuristics finds more usability/UX problems that have higher severity scores than the

control heuristics.

5 Number of identified usability/UX problems that qualify as more critical

Finally, the new set of heuristics is an effective instrument if it finds more usability/UX problems that qualify as the most critical. If a problem occurs less frequently, then it rarely occurs in the application. This aspect implies that if the problem is also severe, it does not significantly influence the use of the application. When assessing criticality, the frequency and severity of a problem are considered to determine how critical the problem is.

The rating scale for criticality ranges from 0 to 8 (see Table 10). The criticality is obtained by summing the frequency and severity value of the problem. In this sense, it is important to identify those problems that have a criticality value that is greater than 4, as this is the average value on the scale of criticality. If the researcher decides to use another rating scale for criticality, then the average value of that scale should be used in the formula that is presented below. The effectiveness in terms of the number of identified usability/UX problems that qualify as more critical (ESC) can be represented as a percentage by using the following Eq. (5):

$$\text{ESC} = \frac{\text{NPC}}{\text{TP}} \times 100 \quad (5)$$

where:

- ESC: effectiveness;
- NPC: number of usability/UX problems identified qualified with a criticality greater than 4;
- TP: total usability/UX problems identified.

Two effectiveness values should be calculated:

Table 12

How different authors validate a new set of heuristics through expert judgment.

Set of proposed heuristics	Participants selected	Elements evaluated
UX heuristics for national park websites [11]	3 experts performing a heuristic evaluation (practitioners)	<ul style="list-style-type: none"> – The survey proposed in the methodology was applied [11]. – Heuristics were evaluated concerning 4 dimensions (D1 – Utility, D2 – Clarity, D3 – Ease of use, D4 – Necessity of additional checklist) and 3 questions (Q1 – Easiness, Q2 – Intention, and Q3 – Completeness). – A semi-structured survey was applied.
Usability heuristics for u-learning applications [26]	4 expert evaluators	<ul style="list-style-type: none"> – Heuristics were evaluated concerning 3 dimensions ((D1 – Utility, D2 – Clarity, and D3 – Ease of use) and 3 open questions (Q1 – Easiness, Q2 – Intention, and Q3 – Completeness). – A questionnaire was applied. The model of adoption of methods [45] and a questionnaire proposed in [46] were used to design the survey. – Heuristics were checked concerning 3 dimensions (EUP – Easiness of Use Perceived, UP – Utility Perceived, and UI – and Use Intention). – An interview was conducted with several experts.
Cultural-oriented usability heuristics [28]	12 evaluators	<ul style="list-style-type: none"> – Each heuristic's definition was checked for understandability, clarity and consistency. – A survey was applied.
Smartphone's usability heuristics [30]	Reference [47] does not indicate the number of experts	<ul style="list-style-type: none"> – Heuristics were evaluated concerning 4 dimensions (D1 – Utility, D2 – Clarity, D3 – Ease of use, and D4 – Need of additional evaluation elements – “checklists”). – A survey was applied.
Usability heuristics for transactional websites [33]	27 evaluators [30] (11 evaluators with no previous experience and 16 evaluators with previous experience)	<ul style="list-style-type: none"> – Heuristics were evaluated concerning 4 dimensions (D1 – Utility, D2 – Clarity, D3 – Ease of use, and D4 – Relevance).
Usability heuristics for national park websites [33]	4 experts (practitioners)	

- 1 The effectiveness using problems that were identified only by the group that used the new set of heuristics (ESC1).
- 2 The effectiveness using problems that were identified only by the group that used the control heuristics (ESC2).

The new heuristics perform well when ESC1 is at least equal to (but hopefully greater than) ESC2, that is, when the new set of heuristics finds more usability/UX problems with higher criticality scores than the control heuristics.

Table 11 synthesizes the empirical data that show the effectiveness of the new set of heuristics for national park websites.

4.1.7.2. Validation through expert judgment. The researcher checks the validity of the heuristics by asking experts regarding their appropriateness for evaluating the specific application domain and their perception over the new set of heuristics. For this, the researcher: (1) selects participants for expert judgment (researchers and practitioners) and (2) performs the expert judgment through a survey. The explanation of how to design the survey can be reviewed in [11]. Table 12 shows how different authors validate the set of proposed heuristics through expert judgment.

- **Output:** ⑯ Expert judgment results (survey): utility, clarity, ease of use, need for checklist and comments regarding each heuristic

The researcher applies the survey and interprets the results obtained. To analyze the results, the researcher can use descriptive statistics for dimensions and questions and/or evaluate the relation between the dimensions and questions. If the researcher decides to apply a different survey to the one we propose in [11], then the analysis should be done accordingly. Explanations regarding how to apply the proposed survey and how to interpret the results can be reviewed in [48–50].

Applying surveys allows the researcher to obtain both quantitative and qualitative results. It is important to include open questions, as the expert can provide comments and relevant qualitative information to refine and improve the set of heuristics (for example, definitions that are difficult to understand, heuristics that should be added, refined or

eliminated, and so on).

4.1.7.3. Validation through user tests. The researcher can perform user tests to:

- 1 Check whether the usability/UX problems identified in a heuristic evaluation using the new set of heuristics are real usability/UX problems for users.
- 2 Obtain the perceptions of users regarding a specific topic (such as the relevant features of the application domain).

User tests make it possible to determine why the experimental group did not detect the same problems that the control group detected. The tasks to be performed in the user test should involve those problems not detected by the experimental group. Some of the reasons for the above can be:

- 1 The new heuristics are not properly specified (the experimental group did not find the problems detected by the control group since the heuristics are not well understood, so the heuristics need refining);
- 2 There are no appropriate heuristics to evaluate some kinds of problems (a heuristic is missing to detect usability problems, so a new heuristic must be created);
- 3 For the experimental group, the usability problem detected by the control group was not truly a usability problem.

On the other hand, it is possible that during the user test, new usability problems were not detected in the heuristic evaluations. In this case, it is necessary to analyze why these problems were not identified in the heuristic evaluations (poor heuristic definitions, no heuristic to detect the problem, the evaluators subjectively ignored the problems in the evaluation, and so on). As a researcher, it is necessary to review if there is a heuristic in the proposed new set that covers the problem detected in the user test. If the heuristic exists, the researcher should review the heuristic definition and improve it. If the heuristic does not exist, it must be created. Table 13 shows how different authors validate

Table 13
How different authors validate a new set of heuristics via user tests.

Set of proposed heuristics	Type of user test performed	Users involved	User test objective	User test details	Analysis of results
UX heuristics for national park websites [11]	Co-discovery (complemented with a survey)	Sixteen representative users (undergraduate students of tourism, divided into 8 groups of 2 members in each one)	Validate that the problems identified in the heuristic evaluation are perceived as real problems for the users	Yellowstone National Park website was used in the test. The test was designed based on the usability problems identified only by the control group, and rated with a severity greater than 2 and/or a criticality greater than 4	For each task, the related usability problems and the percentage of fulfillment were detected. Two open questions were asked to identify if the elements considered important for the users are evaluated by the set of heuristics
	Focus group	Sixteen representative users (undergraduate students of tourism, divided into 2 groups of 8 members in each one)	Validate whether the set of heuristics evaluates the features considered relevant by users	The participants discussed the features that a national park website should have. The focus groups lasted approximately 15 min.	The comments obtained were analyzed. Each comment included the associated heuristic that evaluates each comment (if it exists).
	Usability test	Five users	Validate whether evaluators using heuristics subjectively ignored the problems	The test was focused on the 6 usability problems identified only by the control group.	Detect if the problems are in fact perceived as real problems by the users.

the set of proposed heuristics via user tests.

- **Output:** ⑦ User tests results: users' perceptions

The researcher conducts the user test and interprets the results obtained. It is important to analyze both the quantitative and qualitative results. For instance, to develop heuristics for national park websites, the researchers validated the heuristics through co-discovery and focus group results (see Table 13). The researchers analyzed the quantitative and qualitative data.

For the co-discovery tests, the researchers analyzed the percentage of task fulfillment (quantitative results, Table 14) and user comments (qualitative results, Table 15). Based on the results, the investigators identified that it was not necessary to create new heuristics since all user comments were covered by the heuristics.

During the focus group, the researchers analyzed all relevant topics related to the UX in the national park websites (see Table 13). Then, the investigators checked whether each topic was covered by a related heuristic, and based on the results, they determined that all topics were covered by at least one heuristic. This finding means that the set of heuristics for national park websites evaluates the features considered relevant for the users. In addition, new checklist items were added to the two heuristics (HPN3 and HPN7) based on the users' comments.

4.1.8. Step 8: refinement stage

- **Inputs:** ⑤ heuristic evaluation results: effectiveness of heuristics; ⑥ expert judgment results (survey): utility, clarity, ease of use, need for checklist and comments regarding each heuristic; and ⑦ user tests results: users' perceptions.

To perform step 8, the researcher needs at least one of the 3 inputs proposed. Depending on the validations made in step 7, the researcher can have input ⑤, ⑥ and/or ⑦. Importantly, at least one input is available to perform step 8. For instance, to develop heuristics for the national park websites, the researchers used inputs ⑤ and ⑥ in the first iteration and input ⑦ in the second iteration.

In the refinement stage, the researcher refines the new set of proposed heuristics based on the results obtained in the validation stage. He/she defines the heuristics to be created, refined and/or deleted.

In addition, he/she decides if it is necessary to iterate and apply some stages once again. We recommend performing at least two iterations to validate the set of heuristics twice, performing different experiments and using different case studies.

- **Output:** ⑧ refining document: (1) what heuristics to create, refine and/or delete, why, and how to do it; (2) what steps to repeat.

The researcher analyzes the results using the inputs and determines which changes should be made to the heuristics to improve them. To document the refinements, the researcher can use a table including the ID of the heuristic, its name, the problem detected, and how to solve it. More details on how to document the information can be reviewed in [11].

To develop heuristics for the national park websites, the researchers documented the following refinements: (1) make changes in the definition of some heuristics; (2) add more detail to the specification of the heuristics (more items in the template); and (3) eliminate 5 heuristics of the preliminary set and create 1 heuristic (obtaining a final set of 14 refined and validated heuristics in the second iteration) [11]. In addition, these investigators defined what stages to repeat to improve the set of heuristics. The researchers decided to repeat steps 3, 4, 5, 6, 7, and 8 (see Section 4.2).

In [30], to develop heuristics for smartphones, the authors performed the following refinements: (1) implemented changes in the definition of some heuristics; (2) added more detail to the specification of the heuristics (more items in the template); (3) added new heuristics to cover specific aspects of smartphones that had not been considered at the beginning; and (4) changed the ID of the heuristics [30,47,51].

On the other hand, to develop cultural-oriented usability heuristics [28], authors performed the following refinements: (1) completed the specification of each heuristic by adding checklists [52,53]; (2)

Table 14

Co-discovery test quantitative results.

Task	Usability problem related	Percentage of task fulfillment	Description
Find an event at Mammoth Hot Springs	- The website contains static, very theoretical and irrelevant information. - The calendar icon is not very representative	37.5% (3 of 8 groups)	- 4 groups did not recognize the icon for direct access to the calendar - 3 groups navigated through another section since they could not find the information they were looking for - 7 of 8 groups were able to correctly perform the task
Search the audio of winter wolves in the audio section	It is confusing to use the search engine (confusing options).	87.5% (7 of 8 groups)	- 1 of 8 groups could not use the search engine correctly

Table 15

Co-discovery test qualitative results.

Question	Comment	Number of users who provided comments	Heuristic associated
Q1: What did you like most about the website?	The multimedia resources (images, videos and audios)	10 of 16 users (62.5%)	NPH2 (Multimedia resources)
	The complete information of the park (activities, parks, animals, seasons, and so on)	8 of 16 users (50%)	NPH3 (Information of interest)
Q2: What did you like the least about the website?	The website design	6 of 16 users (37.5%)	NPH11(Aesthetic and minimalist design)
	The difficulty to find specific information (events, prices, itineraries, and so on)	6 of 16 users (37.5%)	NPH3 (Information of interest)
	The lack of an option to change the language.	7 of 16 users (43.75%)	NPH4 (Match between the system and the real world)

changed the specification of the heuristics (ID changes, definition improvements); and (3) eliminated a heuristic from the preliminary set (13 heuristics, [52,53]), obtaining a final set of 12 refined and validated heuristics [28].

4.2. Iterating the methodology stages

It is possible to apply a methodology that follows the 8 stages and complete the entire process without iterations. However, we recommend applying the methodology iteratively and performing at least two iterations to refine and improve the set of heuristics. Satisfactory results are difficult to obtain in a single iteration. To start the second iteration, output @ is used. From the second iteration, not all inputs are mandatory in the stages. The inputs will depend on the activities carried out and the outputs obtained. We recommend using all stages in the first iteration (eventually, step 2 may be optional). Subsequent iterations include fewer stages because they are oriented toward validation and refinement.

On the other hand, we recommend using more than one method to validate and compare the results that are obtained by different methods. We recommend always validating the set of heuristics through heuristic evaluation. The choice of methods will depend on the available resources (e.g., evaluators, experts, and time). The researcher must choose the most appropriate method for the validation (heuristic evaluation, expert judgment, user tests, surveys, focus group, etc.).

The researcher can end the process of creating heuristics if he/she considers that after several iterations and validations, the heuristics are well designed. However, it is difficult to say exactly when the process should stop. A set of heuristics can always be improved. In addition, the rapid evolution of technologies/applications can render a set of heuristics obsolete.

Fig. 2 shows the number of iterations performed to develop heuristics for national parks (diagram a), smartphones (diagram b) [30], and cultural aspects (diagram c) [28]. The iteration number is marked in the diagrams as “It. n”.

To develop heuristics for the national park websites, researchers performed two iterations using our methodology (see diagram a, Fig. 2) [11]. These investigators performed all stages in the first iteration except step 2 and repeated the stages from 3 to 8 in the second iteration. In the second iteration, the researchers repeated the process from step 3

since in the refining stage of the first iteration, they detected new specific aspects that were not covered by the set of heuristics. For step 7 (validation stage), it was relevant to perform experiments with experts in the area of application (tourism students) via user tests. The participants had experience in the domain and could provide useful information for the heuristic validation.

To develop heuristics for smartphones [30,47,51], the authors performed five iterations using the methodology proposed by Rusu et al. [43] (see diagram b, Fig. 2). These researchers performed stages from 1 to 4 in the first iteration to specify the new set of heuristics. Iterations 2, 3, 4 and 5 were used to validate and refine the heuristics, and they performed several experiments for the validation (heuristic evaluations and expert opinion). In the last iteration (iteration 5), they specified the final version of the heuristics.

To develop cultural-oriented heuristics [28,52,53], the authors performed four iterations using the methodology proposed by Rusu et al. [42] (see diagram c, Fig. 2). These researchers performed stages from 1 to 4 in the first iteration to specify the new set of heuristics. Iterations 2, 3 and 4 were used to validate and refine the heuristics. These authors performed several experiments for the validation (heuristic evaluations and expert opinion), and in the last iteration (iteration 4), they specified the final version of the heuristics.

5. Conclusions

The large number of sets of usability/UX heuristics that have been developed to date (more than 80, according to [20,21]) reflect the need to create instruments that allow the effective evaluation of specific application domains (and their specific features). We proposed a formal methodology for establishing new sets of heuristics [11] to facilitate the process of developing usability/UX heuristics. This methodology includes 8 stages. Each stage of the methodology has defined inputs and outputs and details the activities to be performed to create the heuristics. The methodology is flexible and provides the researcher with the possibility of deciding which stages to carry out, how many iterations to perform and which stages to repeat, based on the specific application domain and the available resources (evaluators, experts, and time). It is possible to repeat some (or all) stages, omit stages, and decide which inputs to use in each iteration.

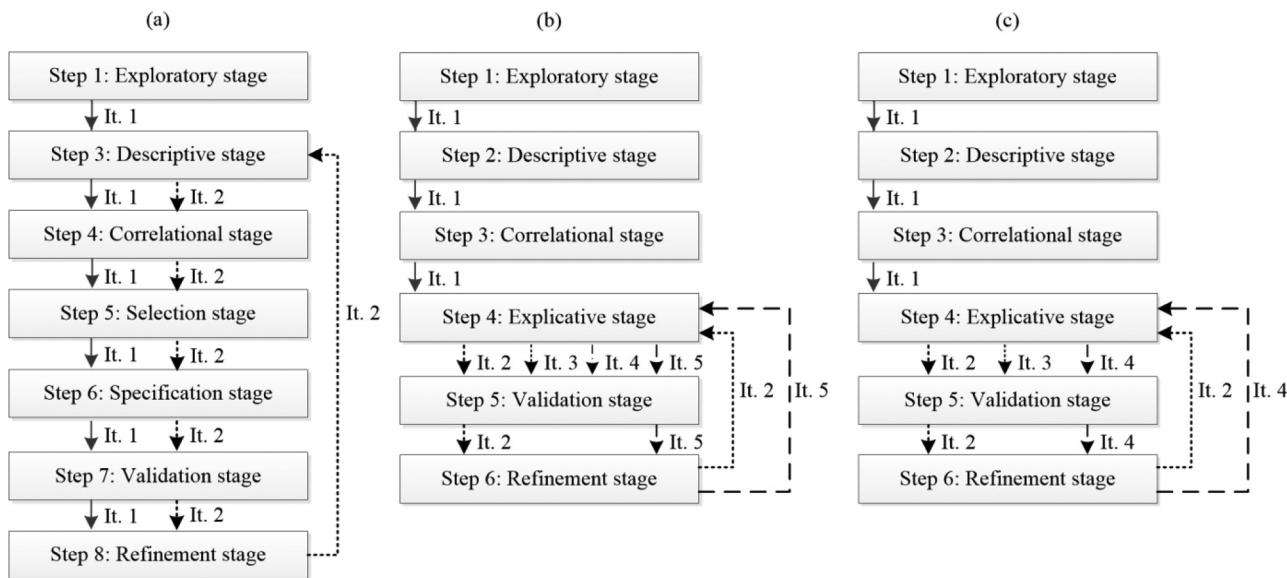


Fig. 2. Iterations performed to develop new sets of heuristics. (a) UX heuristics for national park websites (using the methodology proposed by Quiñones et al. [11]); (b) Usability heuristics for smartphones [30] (using the methodology proposed by Rusu et al. [43]); and (c) Cultural-oriented usability heuristics [28] (using the methodology proposed by Rusu et al. [43]).

As mentioned in [21], developing heuristics should not stop once the heuristics are proposed. It is necessary to perform a robust and rigorous validation in which the effectiveness of the heuristics for specific domains is compared to other heuristics (general and/or specific heuristics). Additionally, standard measures should be adopted to indicate the heuristics' effectiveness. We propose several quantitative and qualitative validation methods in the methodology [11], including precise criteria to evaluate the effectiveness of the heuristics.

To help the experts and/or researchers during the process of applying the methodology, we suggest how to perform each activity, step by step, and how to validate the new heuristics. We also advise how to perform iterations, especially to refine and improve the proposed set. We use several case studies to explain how to apply the methodology. We hope that this detailed explanation will serve as a guide for experts and/or researchers in the creation of new heuristics. We expect that this guide will allow them to see more clearly and concretely what they need to start each stage, how to perform the activities and what they should obtain at the end.

In future work, we plan to fully explain the development of a new set of heuristics for a specific domain by applying the methodology. In addition, we would like to apply the methodology to develop heuristics for attributes other than usability/UX. We are particularly interested in applying the methodology to develop Customer eXperience heuristics [54,55]. If necessary, the methodology will be adapted. We also want to examine how certain stages of the methodology (or possibly all stages) would benefit when using supporting software tools.

Conflicts of interest

None.

Acknowledgments

The authors would like to thank all the participants (experts and researchers) who were involved in the experiments for this study, especially the members of the “UseCV” Research Group in Human-Computer Interaction (School of Informatics Engineering, Pontificia Universidad Católica de Valparaíso – Chile). We would also like to thank the experts Dania Delgado and Daniela Zamora who developed the heuristics for the national park websites for sharing their work. This

work was supported by the Universidad de Playa Ancha, Valparaíso, Chile; we would like to thank Prof. Dr. Virginica Rusu and her students.

References

- [1] A. Anganes, M.S. Pfaff, J.L. Drury, C.M. O'Toole, The heuristic quality scale, *Interact. with Comput.* 28 (5) (2016) 584–597.
- [2] J. Nielsen, R. Molich, Heuristic evaluation of user interfaces, *Proceeding of Conference on Human factors in Computing Systems, SIGCHI '90*, 1990, pp. 249–256.
- [3] J. Nielsen, “Ten Usability Heuristics”, <https://www.nngroup.com/articles/ten-usability-heuristics/> (accessed 23 November 2018).
- [4] C. Rusu, S. Roncaglio, V. Rusu, C. Collazos, A methodology to establish usability heuristics, *Proceedings of the Fourth International Conference on Advances in Computer-Human Interactions, ACHI2011*, 2011, pp. 59–62.
- [5] D. Van Greunen, A. Yeratziotis, D. Pottas, A three-phase process to develop heuristics, *Proceedings of the 13th ZA-WWW Conference, Johannesburg*, 2011.
- [6] M. Hub, V. Čapková, et al., Heuristic evaluation of usability of public administration portal, in: Narsingh Deo, et al. (Ed.), *Applied Computer Science, Proceedings of International Conference on Applied Computer Science, ACS*, 2010.
- [7] F. Franklin, F. Breyer, J. Kelner, Heurísticas de usabilidad para sistemas colaborativos remotos de realidad aumentada, *Proceedings of XVI Symposium on Virtual and Augmented Reality, SVR*, 2014, pp. 53–62.
- [8] B. Lechner, A. Frühling, S. Petter, H. Siy, The chicken and the pig: user involvement in developing usability heuristics, *Proceedings of the Nineteenth Americas Conference on Information Systems*, 2013.
- [9] S. Hermawati, G. Lawson, A user-centric methodology to establish usability heuristics for specific domains, *Proceedings of the International Conference on Ergonomics & Human Factors*, 2015, pp. 80–85.
- [10] A.R. Hevner, S.T. March, J. Park, S. Ram, Design science in information systems research, *J. MIS Q.* 28 (1) (2004) 75–105.
- [11] D. Quiñones, C. Rusu, V. Rusu, A methodology to develop usability/user experience heuristics, *Comput. Stand. Interfaces* 59 (2018) 109–129.
- [12] D. Delgado, D. Zamora, *Experiencia del Usuario en Sitios Web de Parques Nacionales*, Undergraduate Thesis Pontificia Universidad Católica de Valparaíso, Chile, 2017.
- [13] V. Roto, M. Lee, K. Pihkala, B. Castro, A. Vermeeren, E. Law, K. Väänänen-Vainio-Mattila, J. Hoonhout, M. Obrist, All About UX, *Information for User Experience Professionals* (2018), <http://www.allaboutux.org> (Accessed 15 November).
- [14] ISO 9241-210, *Ergonomics of Human-system Interaction — Part 210: Humancentred Design for Interactive Systems*, International Organization for Standardization, 2010.
- [15] T. Jokela, N. Iivari, J. Matero, M. Karukka, The standard of user-centered design and the standard definition of usability: analyzing ISO 13407 against ISO 9241-11, *Proceedings of the Latin American Conference on Human-computer interaction, ACM*, 2003, pp. 53–60.
- [16] P. Morville, “User Experience Design”, 2004. http://semanticstudios.com/user_experience_design/ (Accessed 15 November 2018).
- [17] M. Revang, “The User Experience Wheel”, 2007. <http://userexperienceproject.blogspot.com/2007/04/user-experience-wheel.html> (Accessed 15 November 2018).

- 2018).
- [18] L. Arhipainen, M. Thäti, Empirical evaluation of user experience in two adaptive mobile application prototypes, Proceedings of the 2nd International Conference on Mobile and Ubiquitous Multimedia, 2003, pp. 27–34.
- [19] J. Garret, *The Elements of User Experience: User-Centered Design for the Web and Beyond*, Pearson Education, 2010.
- [20] D. Quiñones, C. Rusu, How to develop usability heuristics: a systematic literature review, *Comput. Stand. Interfaces* 53 (2017) 89–122.
- [21] S. Hermawati, G. Lawson, Establishing usability heuristics for heuristics evaluation in a specific domain: is there a consensus? *Appl. Ergon.* 56 (2016) 34–51.
- [22] C. Rusu, S. Roncagliolo, G. Tapia, D. Hayvar, V. Rusu, D. Gorgan, Usability heuristics for grid computing applications, Proceedings of the Fourth International Conference on Advances in Computer-Human Interactions, ACHI, 2011, pp. 53–58.
- [23] J. Nielsen, “Usability 101: Introduction to Usabilityusability”, 2012. <https://www.nngroup.com/articles/usability-101-introduction-to-usability/>. (Accessed 15 November 2018).
- [24] L. Masip, M. Oliva, T. Granollers, User experience specification through quality attributes, *Hum.-Comput. Interact. – Interact.* (2011) 656–660.
- [25] N. Aguirre, Experiencia de Usuario en Museos Virtuales, Undergraduate Thesis Pontificia Universidad Católica de Valparaíso, Chile, 2015.
- [26] F. Sanz, R. Gálvez, C. Rusu, S. Roncagliolo, V. Rusu, C.A. Collazos, J.P. Cofré, A. Campos, D. Quiñones, A set of usability heuristics and design recommendations for u-learning applications, *Information Technology: New Generations*, Springer International Publishing, 2016, pp. 983–993.
- [27] J.P. Cofré, Usabilidad en u-Learning, Master Thesis Pontificia Universidad Católica de Valparaíso, Chile, 2013.
- [28] J. Díaz, C. Rusu, C.A. Collazos, Experimental validation of a set of cultural-oriented usability heuristics: e-Commerce websites evaluation, *Comput. Stand. Interfaces* 50 (2017) 160–178.
- [29] R. Meszaros, I. Lagzi, Z. Barcza, G. Gelybo, *GreenView Application Specifications*, Eotvos Lorand University, Hungary, 2008.
- [30] R. Inostroza, C. Rusu, S. Roncagliolo, V. Rusu, C.A. Collazos, Developing SMASH: a set of SMARTphone's usability Heuristics, *Comput. Stand. Interfaces* 43 (2016) 40–52.
- [31] A. Campos, C. Rusu, S. Roncagliolo, F. Sanz, R. Gálvez, D. Quiñones, Usability heuristics and design recommendations for driving simulators, *Information Technology: New Generations*, Springer International Publishing, 2016, pp. 1287–1290.
- [32] D. Quiñones, C. Rusu, S. Roncagliolo, Redefining usability heuristics for transactional web applications, *Information Technology: New Generations (ITNG)*, 2014 11th International Conference on, IEEE, 2014, pp. 260–265.
- [33] D. Quiñones, C. Rusu, S. Roncagliolo, V. Rusu, C.A. Collazos, Developing usability heuristics: a formal or informal process? *IEEE Latin Am. Trans.* 14 (7) (2016) 3400–3409.
- [34] Real Academia Española, “Definición de parque nacional”, 2014. <https://dle.rae.es/?id=RyGZA0Z> (Accessed 11 March, 2019).
- [35] L. Bonastre, T. Granollers, A set of heuristics for user experience evaluation in e-Commerce websites, 7th International Conference on Advances in Computer-Human Interactions, IARIA, 2014, pp. 27–34.
- [36] G.H. Hofstede, M. Minkov, *Cultures and Organizations: Software of the Mind*, McGraw-Hill Professional, 2001, pp. 1–29.
- [37] L. Smith, T. Dunckley, S. French, Minocha, Y. Chang, A process model for developing usable cross-cultural websites, *Interact. Comput.* 16 (1) (2004) 63–91.
- [38] K.a Huggins, B.B. Holloway, D.W. White, Cross-cultural effects in E-retailing: the moderating role of cultural confinement in differentiating Mexican from non-Mexican Hispanic consumers, *J. Bus. Res.* 66 (3) (2013) 321–327.
- [39] L. Sangwon, R. Koubeekb, The effects of usability and web design attributes on user preference for e-Commerce web sites, *Comput. Ind.* 61 (4) (2010) 329–341.
- [40] F. Paz, F.A. Paz, J.A. Pow-Sang, L. Collantes, Usability heuristics for transactional web sites, Proceedings of the 11th International Conference on Information Technology: New Generations, 2014, pp. 627–628.
- [41] Korhonen, E.M.I. Koivisto, Playability heuristics for mobile games, Proceedings of Mobile Human-Computer Interactions, 2006, pp. 9–16.
- [42] M. Omar, A. Jafar, Heuristics evaluation in computer games, Proceedings of Information Retrieval & Knowledge Management, 2010, pp. 188–193.
- [43] C. Rusu, S. Roncagliolo, V. Rusu, C.A. Collazos, A methodology to establish usability heuristics, Proceedings of the Fourth International Conference on Advances in Computer-Human Interactions, ACHI, 2011, pp. 59–62.
- [44] Nielsen, “Severity ratings for usability problems”, 1995. <https://www.nngroup.com/articles/how-to-rate-the-severity-of-usability-problems/> (Accessed 20 November 2018).
- [45] D.L. Moody, Dealing With Complexity: A Practical Method for Representing Large Entity Relationship Models, University of Melbourne, Department of Information Systems, 2001.
- [46] N.C. Fernández, Un Procedimiento de Medición de Tamaño Funcional para Especificaciones de Requisitos (2006) Doctoral Thesis.
- [47] R. Inostroza, C. Rusu, S. Roncagliolo, V. Rusu, Usability heuristics for touchscreen-based mobile devices: update, Proceedings of the 2013 Chilean Conference on Human-Computer Interaction, ACM, 2013, pp. 24–29.
- [48] C. Rusu, V. Rusu, S. Roncagliolo, D. Quiñones, V.Z. Rusu, H.M. Fardoun, D.M. Alghazzawi, C.A. Collazos, Usability heuristics: reinventing the wheel? Proceedings of the International Conference on Social Computing and Social Media, Springer International Publishing, 2016, pp. 59–70.
- [49] V. Rusu, C. Rusu, D. Quiñones, S. Roncagliolo, C.A. Collazos, What happens when evaluating social media's usability? Proceedings of the International Conference on Social Computing and Social Media, Springer, 2017, pp. 117–126.
- [50] V. Rusu, C. Rusu, D. Guzmán, S. Roncagliolo, D. Quiñones, Online travel agencies as social media: analyzing customers' opinions, Proceedings of the International Conference on Social Computing and Social Media, Springer, 2017, pp. 200–209.
- [51] R. Inostroza, C. Rusu, S. Roncagliolo, C. Jiménez, V. Rusu, Usability heuristics for touchscreen-based mobile devices, Proceedings of the 9th International Conference on Information Technology: New Generations, 2012, pp. 662–667.
- [52] C.Rusu Diaz, J.A. Pow-Sang, S. Roncagliolo, A cultural-oriented usability heuristics proposal, Proceedings of the 2013 Chilean Conference on Human-Computer Interaction, ACM, 2013, pp. 82–87.
- [53] J. Diaz, C. Rusu, A. Pow-Sang, S. Roncagliolo, Una Propuesta de Heurísticas de Usabilidad Orientados a Aspectos Culturales, 8CCC – Octavo Congreso Colombiano de Computación (2013).
- [54] C. Bascur, C. Rusu, D. Quiñones, User as customer: touchpoints and journey map, International Conference on Human Systems Engineering and Design: Future Trends and Applications, Springer, Cham, 2018, pp. 117–122.
- [55] V. Rusu, C. Rusu, F. Botella, D. Quiñones, Customer eXperience: is this the ultimate eXperience? Proceedings of the XIX International Conference on Human-Computer Interaction, paper No. 21, ACM, 2018.



Daniela Quiñones was born in Viña del Mar, Chile in 1989. She received the M.S. degree in Computer Science from the Pontificia Universidad Católica de Valparaíso (PUCV), Chile, in 2016 and the PhD degree in Informatics Engineering from the same university in 2018. She is associate professor at PUCV. She has worked in several software companies. Since 2018, she has been dedicated to academia. Her research interests have focused on Human-Computer Interaction (HCI), Customer eXperience, User eXperience, Usability, and Usability Engineering. Dr. Quiñones is member of the “UseCV” Research Group in HCI at PUCV, Chile and member of ACM-SIGCHI.



Cristian Rusu was born in Dej, Romania in 1960. He received the PhD degree in Applied Informatics from the Technical University of Cluj-Napoca, Romania, in 2002. He is full professor at Pontificia Universidad Católica de Valparaíso (PUCV) in Chile. He has worked in several software companies and research institutes in Romania and Chile. Since 1999, he has been fully dedicated to academia. He serves on editorial boards and conference program committees and disseminates User eXperience importance in Chile and Latin America. His research interests are focused on Human-Computer Interaction (HCI), User eXperience, Usability, Customer eXperience, and Service Science. As the former Chair of the Chilean ACMSIGCHI Chapter, he is also the head of the “UseCV” Research Group in HCI at PUCV.



Hedonic and Ergonomic Quality Aspects Determine a Software's Appeal

Marc Hassenzahl, Axel Platz, Michael Burmester and Katrin Lehner

Corporate Technology - User Interface Design, Siemens AG

81730 Munich, Germany

+49 (0) 89 636-49653

marc.hassenzahl@mchp.siemens.de

"we continue to see [...] the prospect of a decade of research analysis of usability possibly failing to provide the leverage it could on designing systems people will really want to use by ignoring what could be a very potent determination of subjective judgements of usability - fun" (p. 23) [6].

ABSTRACT

The present study examines the role of subjectively perceived *ergonomic quality* (e.g. simplicity, controllability) and *hedonic quality* (e.g. novelty, originality) of a software system in forming a judgement of appeal. A hypothesised research model is presented. The two main research question are: (1) Are ergonomic and hedonic quality subjectively different quality aspects that can be independently perceived by the users? and (2) Is the judgement of appeal formed by combining and weighting ergonomic and hedonic quality and which weights are assigned?

The results suggest that both quality aspects can be independently perceived by users. Moreover, they almost equally contributed to the appeal of the tested software prototypes. A simple averaging model implies that both quality aspects will compensate each other.

Limitations and practical implication of the results are discussed.

Keywords

perceived software quality, emotional usability, hedonic components, joy of use

INTRODUCTION

In 1988, Carroll and Thomas [6] admonished us not to confuse the concepts *easy to use* and *fun to use* when talking about software quality. They argued that ease of use implies simplicity, which in turn is partly incompatible with fun. By making something as simple as possible, there is a good chance to make it boring as well. On the other hand, fun requires a subtle balance of not being too simple and not being too challenging (see [7]). As Carroll and Thomas put it: "we do not necessarily want to merely make things as simple as possible. We ought to make them fun" (p. 22).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CHI '2000 The Hague, Amsterdam
Copyright ACM 2000 1-58113-216-6/00/04...\$5.00

They call for the scientific study of fun and specialised design methods for promoting fun.

Since then *fun of use* did not receive that much attention in the field of Human-Computer Interaction. It plays a minor role in Technology Acceptance literature. For example, Davis et al. [8] showed that perceived fun (defined as "the extent to which using a software system is enjoyable in its own right") can accelerate usage intention if the software system is already perceived as useful ("the extent to which a person believes that using a particular software system would enhance his or her job performance"). Fun had no effect on usage intentions if a software system was not regarded as useful.

Igbaria, Schiffman and Wieckowski [12] studied the impact of perceived usefulness and perceived fun on both system usage and user satisfaction in a work context. Their results showed an almost equal effect of perceived fun and perceived usefulness on system usage. Perceived fun had even a stronger effect on user satisfaction than perceived usefulness. User satisfaction in turn had a clear effect on system usage. It can be concluded that enhancing perceived fun will primarily lead to increased time spent with a software system. This in turn may lead to a better understanding and/or a more productive use of the system. Furthermore, enhancing perceived fun should be a valuable road to directly increase user satisfaction.

Both studies succeed in demonstrating the positive impact of perceived fun - a factor independent of the efficiency and effectiveness of a software system - on usage intentions and user satisfaction. What these studies do not provide is an idea which design features will increase perceived fun.

One approach to find design principles which have the power to promote fun/enjoyment of a software system is to analyse what makes computer games fun [6;19;20]. Malone [19] identifies the following three broad design categories: "Challenge", "Fantasy" and "Curiosity". Each consists of several principles and recommendations for designing an appealing computer game. Among those, principles both consistent and contradictory to the notion of usability can be found. For example, the recommendation "provide a fantasy" is generally consistent with the idea of using a "metaphor" to increase familiarity and thereby the usability of a system.

However, besides the cognitive aspects (increased familiarity) Malone stresses the emotional aspects of a metaphor used, i.e. its power to satisfy the user's emotional needs. Mainly contradictory to usability is the principle to foster curiosity by designing a system that is novel and surprising (but not completely incomprehensible). In order to work, novelty and surprise must impair at least the external consistency of the software, a core principle of usability. In usability design, this might conflict with the task-related efficiency and effectiveness of the software system. This raises the question, whether it is appropriate to call for enjoyment and fun when it comes to work-related software systems (e.g. [11]).

The strong focus on task-related efficiency and effectiveness arises from the implicit notion that the computer is a tool - or even stronger - *has to be* a tool when applied at the workplace. This somewhat conservative perspective where "things must be taken seriously" is predominant in business and might be at least partly wrong considering the role of perceived fun/enjoyment in the context of technology acceptance at the workplace. The somewhat narrow focus of usability on task-related efficiency and effectiveness is eventually criticised in studies concerning consumer products [1;14;18].

What is needed is an expanded concept of usability which adopts enjoyment and satisfaction of the user as the major design goal¹. Obviously, in a work context designing for efficiency and effectiveness (i.e. traditional usability) will be still a valuable way to reach the goal. Nevertheless, something should be added which makes the software system interesting, novel, surprising etc. Being both usable and interesting, a software system might be regarded as appealing and as a consequence the user may enjoy using it. Such an expanded perspective on usability would take us a step further toward designing *user experiences* [16] instead of merely making a software usable.

An expanded concept of usability was suggested by Logan [17]. He has developed a two-component usability concept that consists of behavioural and emotional usability. Behavioural usability refers to more or less traditional usability. Emotional usability refers "to the degree to which a product is desirable or serves a need(s) beyond the traditional functional objective" (p. 61). Although important, the concept of emotional usability still leaves many questions unanswered. For example, Logan provides no model or data

about whether and how behavioural and emotional usability influence each other.

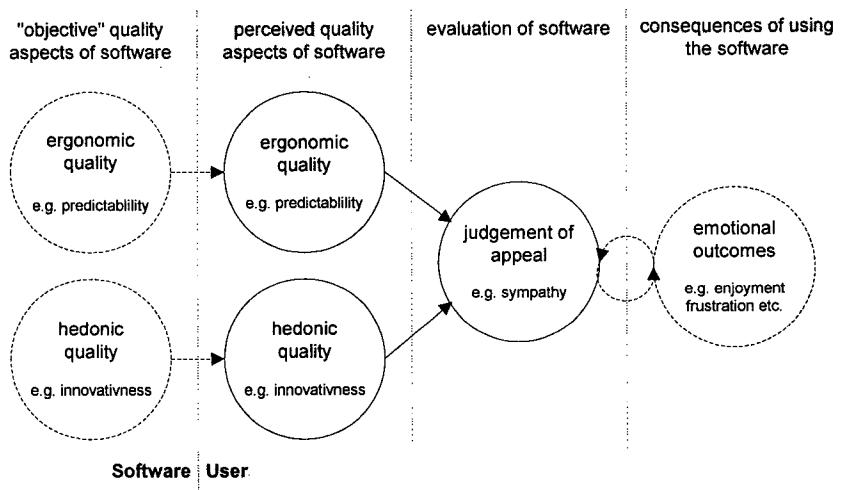
Another line of research to broaden the traditional usability concept mainly considers the impact of visual design on the expected (apparent) usability of a system *before* actually using it [5;15;22]. By demonstrating the impact of visual design on expected (apparent) usability, these studies succeed in establishing visual design as a key design factor. However, this is rather done by regressing visual design on the traditional usability concept than expanding usability itself.

This paper attempts to provide and test a model for an expanded concept of usability that incorporates key factors for designing appealing, enjoyable software interfaces and systems.

APPEALING SOFTWARE SYSTEMS: A HYPOTHESIZED MODEL

Figure 1 shows the elements of the hypothesised model for appealing software systems, i.e. quality aspects of the software system, evaluation of the system and consequences of using it.

Fig. 1: Research Model



A software system can be described on different quality dimensions such as its predictability, controllability etc. The model distinguishes between two groups of quality dimensions (i.e. aspects): *ergonomic* and *hedonic* quality.

Ergonomic quality (EQ) comprises quality dimensions that are related to traditional usability, i.e. efficiency and effectiveness (e.g. [13]). EQ focuses on task-related functions or design issues.

Hedonic quality (HQ) comprises quality dimensions with no obvious relation to the task the user wants to accomplish with the system, such as originality, innovativeness, beauty etc. Although not task-related, the users may regard HQ as an important quality aspect for its own sake.

¹ We are aware that user satisfaction is a part of the usability concept provided by ISO 9241-11. However, it seems as if satisfaction is conceived as a consequence of user experienced effectiveness and efficiency rather than a design goal in itself (see [10] for an example). This implies that assuring efficiency and effectiveness alone guarantees user satisfaction.



Fig. 2: Prototypes

Whether a software system is viewed as appealing by the users or not depends heavily on the users' perception of the quality aspects. For example, "objective" usability (i.e. inherent or intended by the software designer) is not always also perceived by the users [15]. Clearly, a software system designed to be as simple as possible fails if this simplicity can not be perceived or experienced by the users. For this reason, we focus on the subjective perceptions of EQ and HQ provided by the users.

The user's *judgement of appeal* (APPEAL) differs from the mere quality aspects of an additional evaluation. The judgement may be formed by weighting and combining different perceptions of the system's quality aspects (EQ and HQ, respectively).

Emotional outcomes are conceptualised as a consequence of using a software system, rather than as a design goal in itself. If the software system is appealing, the user may experience enjoyment or even fun (if not, she may experience frustration or anger). For this reason, the emotional outcome is not explicitly included in the present study. However, it remains important and should be included in further studies.

The present paper will see to the following research questions.

Q1: Are ergonomic (EQ) and hedonic quality (HQ) subjectively different quality aspects that can be independently perceived by the users?

Q2: Is the judgement of appeal (APPEAL) formed by combining and weighting EQ and HQ? Which weights are assigned to EQ and HQ?

Q3: The research concerning apparent usability [5;15;22]

emphasises user expectations concerning the quality aspects of a software system *before* actually using it. Based on this general idea, we additionally investigated EQ, HQ and APPEAL under both conditions - before and after using a software system. Furthermore, the stability of EQ, HQ and APPEAL will be examined.

METHOD

Participants

Twenty individuals (6 women, 14 men) participated in the study. They were recruited among the employees of Siemens Corporate Technology in Munich. The sample's mean age was 32.5 years (Min 25, Max 57). Computer expertise varied from moderate to high.

Prototypes

Seven software prototypes were designed and implemented (see Figure 2). The participant's task was to switch off a pump in an assumed industry plant. It was a very simple, but realistic task from a domain unknown to the participants. Thereby, unwanted comparison to existing software systems should be reduced. Each prototype allowed the user to accomplish the same task but strongly varied in design and interaction style. The prototypes were designed and implemented by students of visual, industrial and ergonomic design. Multiple design dimensions were varied (e.g. colours, design style). Six out of the seven prototypes had animated parts. The predominant design principle was heterogeneity, i.e. maximising variance. Hence, flaws in the visual and ergonomic design were not corrected.

Semantic differential

A semantic differential (see [9] p. 73) was constructed to measure EQ, HQ and APPEAL. It consists of 23 seven-point scale items with bipolar verbal anchors (see Table 1).

Each scale item was selected beforehand to represent a facet of the quality aspect to be measured (EQ, HQ) and the judgement of appeal (APPEAL) according to the definitions given above. Informal expert reviews were conducted to assure measurement quality of the differential's initial version. The scales were presented in random order.

Pre and Post Usage

All measurements were obtained twice: before and after using the software system. EQ, HQ and APPEAL values obtained *before* actual use are named *expected* (i.e. expected by the user); EQ, HQ and APPEAL values obtained *after* using the system are called *experienced*.

Procedure

The study was carried out in the usability laboratory of the User Interface Design department of Siemens Corporate Technology. Each participant was separately led into the lab by the experimenter. After a short instruction, the experimenter showed the first prototype to the participant for approximately one minute. The prototypes were presented in random order. The participant was then asked to make an initial assessment with the semantic differential before actually using the prototype (*expected* values).

At a signal of the experimenter, the participant had to select the running pump on the screen with the mouse. Depending on the prototype, a question appeared asking whether the participant really wanted to switch off the pump. After a confirmation and a safety check the pump could be turned off. The participant then had to wait until the pump stopped. The end of the task was reached when the participant indicated that s/he was convinced that the pump came to a halt. The whole interaction per prototype took about two minutes.

After finishing the task, the participant was requested to revise his/her initial assessment (*experienced* values). This procedure was repeated for all seven prototypes.

Questionnaires concerning demographics, computer expertise and computer anxiety were handed out. S/He was then shortly debriefed and led out of the laboratory. A session took about an hour.

Tab. 1: Bipolar verbal scale anchors per factor

Scale Item	Anchors	
EQ 1	comprehensible	incomprehensible
EQ 2	supporting	obstructing
EQ 3	simple	complex
EQ 4	predictable	unpredictable
EQ 5	clear	confusing
EQ 6	trustworthy	shady
EQ 7	controllable	uncontrollable
EQ 8	familiar	strange
HQ 1	interesting	boring
HQ 2	costly	cheap
HQ 3	exciting	dull
HQ 4	exclusive	standard
HQ 5	impressive	nondescript
HQ 6	original	ordinary
HQ 7	innovative	conservative
APPEAL 1	pleasant	unpleasant
APPEAL 2	good	bad
APPEAL 3	aesthetic	unaesthetic
APPEAL 4	inviting	rejecting
APPEAL 5	attractive	unattractive
APPEAL 6	sympathetic	unsympathetic
APPEAL 7	motivating	discouraging
APPEAL 8	desirable	undesirable

Note: verbal anchors of the differential are originally in German

RESULTS

Factorial validity and scale reliability of the semantic differential (Q1)

The first question to be answered is whether ergonomic (EQ) and hedonic quality (HQ) are independently perceived by the participants (Q1).

A factor analysis (Principal Components, Varimax rotation) of the *experienced* EQ and HQ items of the semantic differ-

ential extracted two factors with an Eigenvalue higher than 1 (see Table 2). Together both factors explain approx. 68% percent of the total variance. This analysis confirms the initial version of the differential. It shows high consistency with the theoretically assumed factors EQ and HQ. Furthermore, the successfully attained simple structure (e.g. [21]) by applying Varimax rotation shows that EQ and HQ are perceived as independent quality concepts by the participants. For the remainder of the paper, a mean ergonomic and hedonic quality value is computed from the respective single scale item values.

Tab. 2: Factorial validity of experienced EQ and HQ

Scale Item	Principal Components with Varimax	
	Factor 1	Factor 2
EQ 1	.783	
EQ 2	.816	
EQ 3	.715	-.233
EQ 4	.880	
EQ 5	.893	
EQ 6	.824	
EQ 7	.885	
EQ 8	.679	
HQ 1		.805
HQ 2		.731
HQ 3		.722
HQ 4		.854
HQ 5		.892
HQ 6	-.226	.818
HQ 7	-.218	.831
Eigenvalue	5.44	4.76
% Variance explained	36.27	31.72

Note: N=140 (20 participants x 7 prototypes), EQ: ergonomic quality, HQ: hedonic quality, factor loadings < .20 are omitted

An analysis of the *expected* EQ and HQ items revealed similar results.

A factor analysis (Principal Components) of the *experienced* APPEAL items of the semantic differential extracted only one factor with an Eigenvalue higher than 1 (see Table 3).

This confirms the initial selection of items for the APPEAL scale, its univariate character and internal consistency (see scale reliability). For the remainder of the paper a mean judgement of appeal is computed from the values of the single scale items.

An analysis of the *expected* APPEAL items revealed similar results.

Tab. 3: Factorial validity of experienced APPEAL

Scale Item	Principal Components	
	Factor 1	
APPEAL 1		.868
APPEAL 2		.794
APPEAL 3		.683
APPEAL 4		.878
APPEAL 5		.865
APPEAL 6		.903
APPEAL 7		.800
APPEAL 8		.775
Eigenvalue		5.43
% Variance explained		68.20

Note: N=140 (20 participants x 7 prototypes),

APPEAL: judgement of appeal

Table 4 summarises the characteristics of the computed scales.

Tab. 4: Scale characteristics

Scale	Cronbach's Alpha	Mean	Stand. Dev.	Min	Max
<i>expected (pre usage)</i>					
EQ	.93	0.87	1.37	-2.88	3.00
HQ	.93	-0.19	1.47	-3.00	2.75
APPEAL	.95	0.38	1.40	-3.00	2.86
<i>experienced (post usage)</i>					
EQ	.93	0.46	1.50	-2.75	3.00
HQ	.92	0.02	1.40	-2.88	3.00
APPEAL	.93	0.31	1.37	-3.00	3.00

Note: N=140 (20 participants x 7 prototypes), EQ: ergonomic quality, HQ: hedonic quality, APPEAL: judgement of appeal

Predicting the participant's judgement of appeal (Q2)

The judgement of appeal is conceptualised as being formed on the basis of the individual's perceptions of EQ and HQ. To check this assumption two regression analysis were performed in order to predict expected APPEAL from expected EQ and HQ and experienced APPEAL from experienced EQ and HQ (see Table 5).

In both analyses, the variables succeed in predicting APPEAL: EQ and HQ almost equally contribute to the judgement of appeal. No interaction of EQ and HQ was found. This pattern implies an averaging model, with the final judgement of appeal depending on both, the perception of ergonomic and hedonic quality.

Tab. 5: Regression analysis of EQ and HQ on APPEAL (expected and experienced)

Criterion	Adjusted R ²	Predictors	Beta	Std. Error	Sig.
<i>expected (pre usage)</i>					
APPEAL	.74***	EQ***	.51	.05	.000
		HQ***	.62	.04	.000
APPEAL	.74***	EQ***	.40	.14	.003
		HQ***	.53	.10	.000
		EQ x HQ	.16	.03	n.s.
<i>experienced (post usage)</i>					
APPEAL	.67***	EQ***	.64	.05	.000
		HQ***	.58	.05	.000
APPEAL	.67***	EQ***	.64	.12	.000
		HQ***	.72	.14	.000
		EQ x HQ	-.09	.03	n.s.

Note: N=140 (20 participants x 7 prototypes), EQ: ergonomic quality, HQ: hedonic quality, APPEAL: judgement of appeal,

***p<.000

Expected vs. Experienced EQ, HQ and APPEAL (Q3)

Figure 3 shows expected and experienced EQ, HQ and APPEAL (i.e. pre and post usage, see also Table 4 for scale characteristics).

A repeated measurements analysis of variance with the within-subject factors "time of measurement" (pre usage, post usage) and "type of measurement" (EQ, HQ, APPEAL) revealed a highly significant main effect of "type" ($F=11.30$, d.f.=2, p=.000) and a highly significant "type" and "time" interaction ($F=19.92$, d.f.=2, p=.000). No main effect was found for "time" ($F=1.71$, d.f.=1, n.s.).

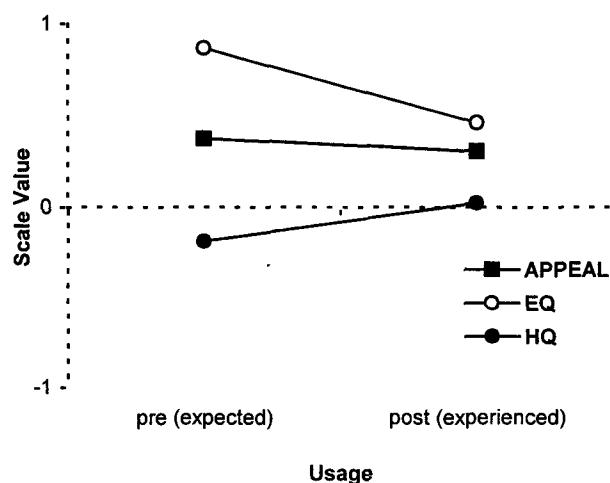


Fig. 2: Expected and experienced EQ, HQ and APPEAL

Single paired sample t-tests for each measure showed a highly significant increase of HQ (diff=.21 scale units,

$t=3.66$, $p=.000$) and a significant decrease of EQ (diff.=-.41 scale units, $t=-3.94$, $p=.000$) after using the prototypes. APPEAL remained stable (diff.=-.07 scale units, $t=-0.87$, n.s.).

Values of expected and experienced EQ, HQ and APPEAL were all significantly correlated (expected EQ - experienced EQ: $r=.64$, $N=139$, $p=.000$; expected HQ - experienced HQ: $r=.89$, $N=139$, $p=.000$; expected APPEAL - experienced APPEAL: $r=.78$, $N=139$, $p=.000$).

DISCUSSION

The results of the factor analysis show that the two assumed quality aspects EQ and HQ can be perceived consistently and independently by users. This is true for both expected and experienced values. In other words, users are able to distinguish task-related aspects from non task-related aspects.

Carroll and Thomas' [6] argument that ease of use implies simplicity, which in turn is partly incompatible with fun, is apparent in the data (see Table 2). The items EQ 3 "simple - complex", HQ 6 "original - ordinary" and HQ 7 "innovative - conservative" are negatively correlated with the opposing aspect (factor). This points to the fact that from a software design perspective it might be impossible to have both aspects maximised. In other words, making it innovative (increase HQ) may result in increased perceived complexity (decreased EQ) and making it simple (increase EQ) may lead to a boring software system (decreased HQ). These findings are consistent with the idea that curiosity can be stimulated by providing an optimal level of informational complexity (e.g. [3]). If the software system is either too simple or too complex, the user will feel bored or overloaded, respectively.

The judgement of appeal (APPEAL) demonstrates to be a highly consistent construct. Even such different dimensions as motivation and aesthetics have the evaluational aspect in common, which is taken into account by the users. The regression analysis of EQ and HQ on APPEAL showed that both quality aspects play an almost equal role in forming the judgement of appeal. Together with the missing interaction, this points at a simple averaging model with EQ and HQ compensating each other. This finding is consistent with the averaging model of Information Integration Theory [2]. This theory proposes that different pieces of information in a multi-attribute judgement are integrated by an averaging process. Taking APPEAL as a multi-attribute judgement and EQ and HQ as internally generated pieces of information (based on expectations or experiences with the software system), the observed averaging is simply a consequence of a cognitive process.

From these results we may cautiously conclude that the hypothesised model for the appeal of software is valuable for guiding future research.

No substantial differences can be found when comparing the way the judgement of appeal is formed before and after using the software. It strikes the eye that the expected

APPEAL seems to be more based on HQ, whereas the experienced APPEAL seems to be based on both HQ and EQ. This may be due to the fact that the HQ quality aspect is much easier to perceive without using the software than EQ.

A comparison of expected and experienced EQ and HQ shows that in general HQ increases whereas EQ decreases. The suggestion of this result is two-fold: First, it shows that HQ and EQ are based on more than simply the appearance of the software system. Both can be influenced by the experiences individuals have with the system. Second, an increase in one quality aspect may always lead to a decrease in the other. This may again be due to the partly incompatibility of HQ and EQ already discussed above.

The fact that APPEAL remains stable might again be due to the averaging process where the decrease in EQ is compensated for by the increase in HQ.

Looking at the correlations of expected with experienced values we find relatively high correlations. This may be a consequence of the relatively short interaction time in our study (see the following section for a discussion).

Limitations

In the following some possible limitations of the present study will be discussed:

Stimulus dependency: The observed effects may depend on the stimulus material provided (the prototypes), hence generalisation may be limited. To reduce this problem beforehand, the prototypes were designed in a way to elicit a wide variety of positive and negative reactions. The scale characteristics (see Table 4) lend support to the notion that this strategy succeeded: minimum and maximum values show that the whole scale was used, the overall means tend to be around the theoretical mean of the scale and the standard deviations for each scale are similar. Nevertheless, there remains the possibility that the presented results can not be generalised to other stimuli. Future studies should address this problem.

Validation of the semantic differential: Internal consistency and factorial validity are important indicators of the reliability of the semantic differential. However, whether the scales really measure the hypothesised quality aspects and the user's evaluation (i.e. the validity of the scales) remains unanswered. Future studies should attempt to validate the scales. For EQ this can be easily achieved by correlating available usability questionnaires. For HQ and APPEAL other ways have to be found.

Lack of judgmental context: The procedure employed in the present study gives no explicit information about the context for the judgement of appeal. The question on hand is whether individuals are able to form a judgement about a software system without thinking of a context, i.e. where to use it. If a context-free judgement is not possible, the participants might simply have induced their own contexts. For example, a group of participants may have thought of how it would be like to use a software system akin to presented prototypes in their daily work. These participants may base

APPEAL more strongly on EQ than on HQ. On the other hand, a second group of participants may have thought that there is a low probability of using a software system akin to prototypes in the future or in their daily work. These participants may emphasise the importance of HQ. The overall result of the two possible strategies would be similar to the observed averaging model with both HQ and EQ being of equal importance.

Future studies should seek to control the context by providing an explicit situation where to apply the software system (e.g. at work). This would change the context-free judgement of appeal to an evaluation that takes the software system's intended "context of use" [4] into account.

Short interaction time: The direction of the expected vs. experienced effect for HQ and EQ (increase of HQ vs. decrease of EQ) seems to be counter-intuitive. We expected an increase in EQ stemming from a sense of reduced complexity and increased familiarity induced by getting to know the system and a decrease in HQ stemming from reduced novelty. The actual results may point at a limitation of the study: the short interaction time. The interaction time of about two minutes might have been too short to change much in the participants' perception and evaluation. The same explanation holds for the relatively strong pre and post usage correlation of HQ, EQ and APPEAL. Further studies should provide more complex systems and should make a longer interaction time (maybe with several measurements over time) possible. This will help to understand how perceptions and evaluations of users change over time.

Practical implications for the design of appealing software systems

Since hedonic quality is perceived by the users and plays a substantial role in forming their judgement of appeal, it should be explicitly taken into account when designing a software system. Without considering hedonic quality a potential source for increased software quality is neglected.

Due to the partial incompatibility of hedonic and ergonomic quality, software designers should try to find a subtle balance of both quality aspects rather than to independently maximise them. Especially interface designers must identify ways to introduce novelty and surprise with their interfaces (and the behaviour of the software system) without sacrificing too much ergonomic quality (e.g. familiarity). From this perspective the impact of hedonic quality on the appeal of a software system may be the rationale for introducing new interface elements (or even completely new metaphors) and to justify the risk of impaired ergonomic quality.

The averaging model implies that a lack of hedonic quality can be compensated by increased ergonomic quality and vice versa. This means also that good usability can be cancelled out by a lack of hedonic quality. Therefore, the primary strategy should be to have both aspects covered. If this is not possible, the designer may concentrate on maximising only one quality aspect. Which one may depend on the software systems purpose and context of use.

CONCLUSION

Despite the possible limitations of the presented study the results look promising and should stimulate further research. The quantitative research approach presented herein should be complemented by a qualitative approach that seeks to identify specific design factors that are able to stimulate the perception of hedonic quality. The major goal should be to provide a better understanding of what makes a software system appealing to people. Such an understanding may guide software design and thus may have positive effects on the acceptance of software, its creative use and the well-being of the users among us who have to spend a large proportion of their professional lives in front of computer screens.

ACKNOWLEDGMENTS

We like to thank the MediaPlant project group, especially Stefan Hofmann, Alard Weisscher, Jochen Klein and Tobias Komischke for designing and implementing the prototypes used in the present study.

REFERENCES

1. Adams, E. and Sanders, E. An evaluation of the fun factor for the Microsoft EasyBall Mouse, in *Proc. of the 39th Human Factors and Ergonomics Society Annual Meeting* (1995), 311-315.
2. Anderson, N. H. *Foundations of information integration theory*. Academic Press, New York, NY, 1981.
3. Berlyne, D. E. Curiosity and exploration. *Science* 153 (1968), 25-33.
4. Bevan, N. and Macleod, M. Usability measurement in context. *Behaviour & Information Technology* 13 1&2 (1994), 132-145.
5. Burmester, M., Platz, A., Rudolph, U., and Wild, B. Aesthetic design - just an add on? in *Proc. of the HCI '99* (1999), 671-675.
6. Carroll, J. M. and Thomas, J. C. Fun. *SIGCHI Bulletin* 19 3 (1988), 21-24.
7. Csikszentmihalyi, M. *Beyond Boredom and Anxiety*. Jossey-Bass, San Francisco, 1975.
8. Davis, F. D., Bagozzi, R. P., and Warshaw, P. R. Extrinsic and Intrinsic Motivation to Use Computers in the Workplace. *Journal of Applied Psychology* 22 14 (1992), 1111-1132.
9. Fishbein, M. and Ajzen, I. *Belief, Attitude, Intention and Behavior*. Addison-Wesley, Reading, MA, 1975.
10. Harrison, A. W. and Rainer, R. K. A general measure of user computing satisfaction. *Computers in Human Behavior* 12 1 (1996), 79-92.
11. Hollnagel, E. Keep cool: The value of affective computer interfaces in a rational world. In *Proc. of HCI International '99* (1999), 676-680.
12. Igbaria, M., Schiffman, S. J., and Wieckowski, T. J. The respective roles of perceived usefulness and perceived fun in the acceptance of microcomputer technology.

- Behaviour & Information Technology* 13 6 (1994), 349-361.
13. ISO. ISO 9241: Ergonomic requirements for office work with visual display terminals. Part 11: Guidance on usability (1996), International Organization for Standardization.
14. Kim, J. and Moon, J. Y. Designing towards emotional usability in customer interfaces - trustworthiness of cyber-banking system interfaces. *Interacting with Computers* 10 (1998), 1-29.
15. Kurosu, M. and Kashimura, K.: Apparent usability vs. inherent usability. In *CHI '95 Conference Companion*. (1995), 292-293.
16. Laurel B. *Computers as Theatre*. Addison-Wesley, Reading, MA, 1993.
17. Logan, R. J. Behavioral and emotional usability: Thomson Consumer Electronics, in M. Wiklund (ed.) *Usability in Practice*. Academic Press, Cambridge, MA, 1994.
18. Logan, R. J., Augaitis, S., and Renk, T. Design of simplified television remote controls: a case for behavioral and emotional usability, in *Proc. of the 38th Human Factors and Ergonomics Society Annual Meeting* (1994), 365-369.
19. Malone, T. W. Toward a theory of intrinsically motivating instruction. *Cognitive Science* 4 (1981), 333-369.
20. Malone, T. W. Heuristics for designing enjoyable user interfaces: Lessons from computer games, in J. C. Thomas and M. L. Schneider (eds.) *Human Factors in Computer Systems*. Ablex, Norwood, NJ, 1984.
21. Thurstone, L. L. Multiple factor analysis: A development and expansion of vectors of the mind. University of Chicago Press, Chicago, 1947.
22. Tractinsky, N. Aesthetics and Apparent Usability: Empirically Assessing Cultural and Methodological Issues, in *Proc. of the CHI '97* (1997), 115-122.

Using the RITE method to improve products; a definition and a case study

Michael C. Medlock, User Testing Lead, Microsoft Games Studios (mmedlock@microsoft.com)

Dennis Wixon, Usability Manager, Microsoft (dennisiwi@microsoft.com)

Mark Terrano, Game Designer, Ensemble Studios (mterrano@EnsembleStudios.com)

Ramon L. Romero, User Testing Lead, Microsoft Games Studios (ramonr@microsoft.com)

Bill Fulton, User Testing Lead, Microsoft Games Studios (billfu@microsoft.com)

ABSTRACT

This paper defines and evaluates a method that some practitioners are using but has not been formally discussed or defined. The method leads to a high ratio of problems found to fixes made and then empirically verifies the efficacy of the fixes. We call it the Rapid Iterative Testing and Evaluation method – or RITE method. Application to the tutorial of a popular game, Age of Empires II, shows this method to be highly effective in terms of finding and fixing problems and generating positive industry reviews for the tutorial.

INTRODUCTION

Traditionally the literature on sample sizes in usability studies has focused on the likelihood that a problem will be found [11, 12, 13, 16, 17, 18, 20]. This literature suggests:

- Running zero participants identifies zero problems.
- The more participants used, the fewer new problems are discovered.
- That calculating the number of participants needed to uncover “enough” problems can be done via a formula based on the binomial probability distribution –but that this number will vary depending on what the experimenter sets as the likelihood of problem detection. It is important to note that this calculation is based on the assumption that the experimenter might see the problem at least once. For example,
 - Observing 4-5 participants will uncover approximately 80% of the problems in a user interface that have a high likelihood of detection (0.31 and higher) [10, 11, 18].
 - Problems in a user interface that do not have a high likelihood of detection (for whatever reason) will require more participants to detect [16, 20].

When the researcher is interested in problems that have a high likelihood of detection, the suggestion has been made that it is more efficient to test with 4-5 users and test more often compared to running fewer, large-sample studies [11, 13].

Depending on the goals and context of the test, there are situations in which running even fewer than 4-5 participants is appropriate and more efficient. Lewis [11] noted that as long as the likelihood of problem detection was very high (0.50 and higher) that 87.5 % of these problems will be uncovered by at least 1 of 3 participants.

However, the usability literature on sample size has often not focused on what we as practitioners view as the primary goal of usability testing in an applied commercial setting: shipping an improved user interface as rapidly and cheaply as possible. We stipulate that when the determination has been made that a discount usability method is appropriate it is more important to get the team to fix problems and to determine the likelihood that a “fix” has solved a problem than to agonize over if every problem has been uncovered. The same likelihood of detection calculation based on the binomial probability distribution can be used for this purpose (and all the same caveats apply). It is noteworthy that relatively few studies have focused on the likelihood that change recommendations will be implemented [7, 15]. A small number of studies have focused on the magnitude of improvement in the user interface of a shipped product or tool, or the relative effectiveness of these improvements in affecting commercial sales or user efficiency [1, 5, 9, 19].

The following are 4 reasons that we encounter that can explain why usability issues that are uncovered do not get fixed:

1. Usability issues are not “believed”. The decision-makers on the product do not think that the issues uncovered are “real” or worthy of a fix.
2. Fixing problems takes time and resources. Development and design resources are scarce, and when faced with the decision between fixing a “working” feature or putting another feature in the product, the choice is often made to add the new feature.
3. Usability feedback arrives late. Feedback that is available when product feature decisions are being made is far more likely to be taken into account than that which arrives after the decisions have already been made. The delay between when a feature is implemented and when usability feedback is delivered to the team is a barrier to those recommendations being used.

4. Teams are uncertain whether a proposed solution will successfully fix the problem. The team doesn't want to undertake a potentially difficult or time-consuming fix if they are not certain to fix the problem. The lack of verification that the fix is working forces the team to implement the usability recommendations on faith, rather than a demonstrated history of accuracy.

This paper defines and evaluates a discount usability method to minimize the 4 problems above, and thus maximize the likelihood that usability work results in getting problems fixed. To promote future discussion and analysis, we have named this method the Rapid Iterative Testing and Evaluation method – or RITE method. The number of participants used is based on how many participants are needed to reasonably determine that an applied fix has solved a problem that was previously uncovered. This method is not “new”, practitioners are already using it [2] but it has not been formally discussed or defined. This method focuses on making very rapid changes to the user interface. More significantly, it evaluates the efficacy of user interface changes immediately after (sometimes within hours) of implementation. What follows is the basics of how to perform the RITE method, and a case study of its use while testing an interactive tutorial for the game Age of Empires II.

WHAT is the RITE Method?

A RITE test is very similar to a “traditional” usability test [4]. The usability engineer and team must define a target population for testing, schedule participants to come in to the lab, decide on how the users behaviors will be measured, construct a test script and have participants engage in a verbal protocol (e.g. think aloud). RITE differs from a “traditional” usability test by emphasizing extremely rapid changes and verification of the effectiveness of these changes. Some of the notable differences are:

- Changes to the user interface are made as soon as a problem is identified and a solution is clear. Sometimes this can occur after observing 1 participant. Once the data for a participant has been collected the usability engineer and team decide if they will be making any changes to the prototype prior to the next participant. In the end the process by which this occurs is up to the usability engineer and team. What follows are some general rules that we found useful for the RITE method:

We classified the issues seen for the participant into four categories:

1. Issues that appear to have an obvious cause and an obvious solution that can be implemented quickly (e.g., text changes, re-labeling buttons, rewording dialog boxes, etc.).
2. Issues that appear to have an obvious cause and an obvious solution that cannot be implemented quickly or within the timeframe of the current test (difficult new features, current features that require substantial design & code changes, etc.).
3. Issues that appear to have no obvious cause and therefore no obvious solution.
4. Issues that may be due to other factors (e.g. test script, interaction with participant, etc).

For each category 1 issue, we implement the fix, and test it with the next participant. For each category 2 issue, we start implementing the fix. As soon as the fix has been implemented, use the revised prototype. For each category 3 and 4 issue, we collect more data to see if they can be upgraded to category 1 or 2 issues, or reclassified as “non” issues.

- There must be agreement prior to the test on tasks every user of the system must be able to perform without exception when using the product. This is critical as it sets the importance of issues seen in the test.
- There must be the ability to make changes very rapidly, (e.g., in less than 2 hours to test with the next participant, or before the next day of testing). This means that development time/resources must be set aside to address the issues raised in the test and the development environment is such that rapid changes can be made.
- Time must be set aside at the end of each participant or day of testing to review results with decision makers and decide if issues raised warrant changes for the next revision of the prototype.
- There must be the ability to run a sufficient number of new participants to verify that the changes made to the user interface have alleviated the issues seen with previous participants and have not caused other unforeseen issues. There is no “set” number for the number of participants needed to verify a fix –the probabilities can be calculated via Lewis’s tables of the binomial probability distribution [10].

Simply meeting the requirements for the method as outlined above will not guarantee its success. To effectively use this method the following conditions must be met:

- The usability engineer must have experience both in the domain and in the problems that users typically experience in this domain. Without this experience it is difficult to tell if an issue uncovered is “reasonably likely to be a problem for other people”. For someone who is inexperienced with a domain a more traditional usability test is more appropriate.
- The decision makers of the development team must participate to address the issues raised quickly (note that this is more than a commitment to attend and review).
- The design team and usability engineer must be able to interpret the results rapidly to make quick and effective decisions regarding changes to the prototype.

CASE STUDY: AGE OF EMPIRES II RITE TEST

PARTICIPANTS

The primary audience for the Age of Empires II tutorial was identified as individuals who had little experience with real-time strategy (RTS) games but who were interested in trying them. The participants were 5 females and 11 males aged 25 to 44 years old. None of the participants had ever played a RTS game but indicated an interest in trying them. All of them had played at least one retail computer game on the PC in the last year.

MEASUREMENT

The primary source of data was the tester's observation of the participants. Specifically:

- Failures. Errors made that resulted in user failure to continue the tutorial—in particular those made on any item identified prior to the test by the team and usability engineer.
- Errors. Errors made that resulted in user confusion—in particular those made on any tasks identified prior to the test.

PROCEDURES

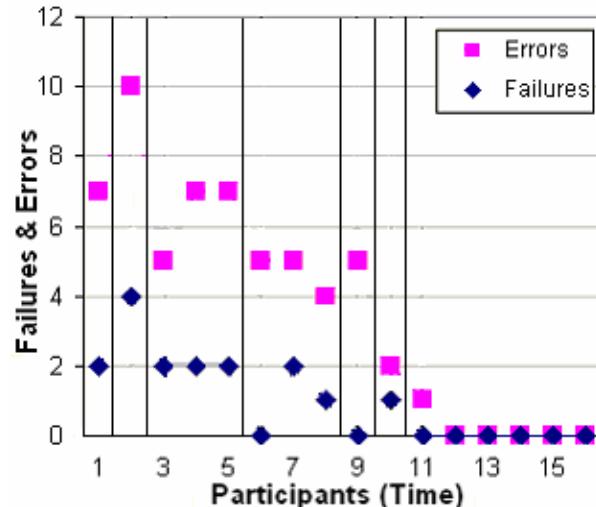
Prior to the testing the usability engineer and the team developed a list of tasks and concepts that the participants should be able to do and/or understand after using the Age of Empires II tutorial. Some of the issues that were identified for which there would be zero error tolerance included: unit movement, performing an “action” on something (gathering resources, attacking, etc.), multi-selection of units, understanding the different kind of resources, showing where resources are accumulating (in the menu areas up top), understanding the “fog of war”, scrolling the main screen with the mouse, using the mini-map, and how to build and repair buildings.

At least one member of the development team (e.g. the Program Manager, Game Designer, Development Lead or User Assistance Lead) was present at every session. After each participant the team would quickly meet with the usability engineer and go over issues seen and place them into one of the 4 categories. For each category 1 issue a fix was implemented, and then the new build was used with the remaining participants. For each category 2 issue a fix was started, and the new build was used with the remaining participants as soon as it was available. This process allowed the team to determine if the “fix” worked or caused new/different problems. For all other issues (category 3 and 4) more data were collected (e.g. more participants run) before any fix was attempted.

AGE OF EMPIRES RITE TEST RESULTS

The results are summarized in Figure 1, which is a record of all the failures and errors over time (as represented by the participants) on the Age of Empires II tutorial. In addition, the graph shows the points at which the tutorial was revised. Every vertical line on the graph indicates a point at which a different version of the tutorial was used. Changes were implemented between participants 1, 2, 5, 8, 9, and 10 (6 iterations).

Figure 1. A record of errors over time as changes to the Age of Empires II tutorial were made using the RITE method.



From Figure 1 it can be seen that the build was changed after the first participant. It is instructive to examine an issue that caused the team to make a fix. In the second part of the tutorial participants are supposed to gather resources with their villagers. One of the resources that they are instructed to gather is wood by chopping trees. However, initially there were no trees on screen and as a result the first participant spent a long time confused as to what to do. The problem was clear and so

was the solution –place some trees within view and teach users how to explore to find trees off-screen. Both of these were done and the issue never showed up again in the next 15 participants.

It's also clear in Figure 1 that the number of failures and other errors generally decreases over time as iterations occur and eventually go to 1 error after the final iteration. Sometimes there were “spikes” after a revision. This was due to the fact that participants could interact with more of the game after previous blocking issues were removed. For example, once the wood chopping issue was removed after the first participant subsequent participants were able to move farther into the tutorial, thus encountering other issues along the way. This illustrates an important strength of the RITE method. By fixing errors early, additional problems can be uncovered.

In addition, we also can calculate the probability that the fixes we made actually fixed the issues we uncovered. After the last participant there were 6 additional participants run with no detection of any previously fixed issue reoccurring (the one issue that turned up never received a fix). Based on Lewis's tables of the binomial probability distribution [10] we are at least 88% likely to have fixed all the issues addressed if their likelihood of occurrence was 0.30 and 47% likely if their likelihood was 0.10. In point of fact the issues that were fixed earlier had an ever greater degree of verification –for example the wood chopping issue fix was verified by 15 participants (almost 100% at 0.30 and 79% at 0.10).

As stated in the introduction, finding problems in a user interface is only half the battle for a practitioner –the other half is getting them fixed. Sawyer, et al [15] proposed a metric they called the “impact ratio” which they used to determine the effectiveness of a test. The impact ratio was defined as:

Problems receiving a fix

$$\text{Impact ratio} = \frac{\text{Problems receiving a fix}}{\text{Total problems found}} * 100$$

Table 1 below shows that the impact ratio was very high using the RITE method.

Table 1. Impact ratio using the RITE method for the Age of Empires II tutorial.

Total problems found	Problems receiving a fix	Impact ratio
31	30	97%

One of the weaknesses of traditional usability tests is that there is often not enough time or resources to discover which “fixes” were not effective. In contrast, the RITE method allows designers to uncover “fixes” that need re-fixing. As such we have come up with a new measure of interest which we will call the “re-fix ratio” and we will define as:

Re-fixes

$$\text{Re-fix ratio} = \frac{\text{Re-fixes}}{\text{Total fixes (including re-fixes)}} * 100$$

Table 2 below shows the re-fix ratio in the RITE test of the Age of Empires II tutorial.

Table 2. Re-fix ratio using the RITE method for the Age of Empires II tutorial.

Total Fixes (includes “re-fixes”)	Changes that needed “re-fixing”	Re-fix ratio
36	6	20%

An example of a re-fix; participants had problems with terminology in Age of Empires II that required repeated fixes. Specifically, participants thought that they needed to convert their villagers units into swordsmen units –when in fact the two unit types were completely unrelated. When the tutorial directed the user to “Train 5 Swordsmen”, participants would move their villagers over to the Barracks (a building that creates swordsmen) assuming that the villagers could be converted into swordsmen. In fact, swordsmen had to be created at the barracks using other resources (e.g. food, gold). When this issue manifested itself, there were inconsistencies in the terminology used in the product. In some places the user interface said “train units” and sometimes it said “create units”. After watching six of eight participants fail to produce new units, the team decided that the text inconsistency was the problem and chose to use the term “train units” throughout the user interface. When the next participant was run the problem reoccurred in the exact same way. As a result the team changed the terminology to “create unit” throughout the rest of the user interface. Once it was “re-fixed” this problem was not seen again after seven participants.

INDEPENDANT reviews and financial success of Age of Empires II

While the RITE method can clearly eliminate user interface problems during a user test, assessing its overall effectiveness for practitioners requires looking at additional information outside of the lab setting, such as team rapport, awards, reviews and sales.

- Mark Terrano –(a lead designer for Age of Empires II) was highly satisfied with the method and wrote about it here, <http://firingsquad.gamers.com/games/aoe2diary/>.

- The Age of Empires II tutorial received an “Excellence” award from Society of Technical Communication in 1999.
- The game play and the tutorial of Age of Empires II received critical acclaim from the gaming press. In many cases the tutorial was singled out as excellent. To see aggregated reviews of Age of Empires II online go to <http://www.gamerankings.com/htmlpages2/804.asp>
- It is practically impossible to relate sales of a product to any single change, feature, or any particular method or technique. However, it is worthwhile noting that Age of Empires II was very successful and that it clearly reached a broader audience than the previous product. Since its release in last August of 1999 until the end of October 2000 Age of Empires II never left the top 10 in games sales according to PC Data. In addition, it continued to break back into the top 10 in 2001 from time to time almost two years after its release. The original Age of Empires was also very successful selling 456,779 units between Oct 1997 – Dec 1998. Age of Empires II sold almost double the number of units of the original Age of Empires in a similar time frame (916,812 copies between Oct 1999 – Dec 2000). The sustained sales of Age of Empires II over the original demonstrates that it is reaching broader markets –many members of which may not have had experience with RTS games before (the segment at which the tutorial was aimed).

CONCLUSIONS

The goals of the RITE method are to identify and fix as many issues as possible and to verify the effectiveness of these fixes in the shortest possible time. These goals support the business reason for usability testing, i.e. improving the final product as quickly and efficiently as possible. The results presented here suggest that at least in the context of its use for the Age of Empires II tutorial, the RITE method was successful in achieving its goals. RITE’s goal is very similar to other iterative methods (cognitive walkthroughs and heuristic reviews [14], GOMS analysis [3], usability tests in general, etc.). It differs from other methods in the rapidity with which fixes occur and are verified. In the introduction we postulated 4 reasons that could explain why usability recommendations do not make it into products. The RITE method did a good job addressing all of these issues in the Age of Empires II case study.

1. The usability issues were “believed”. The decision-makers had often pre-defined what tasks participants should be able to accomplish. In addition, through their constant involvement the decision-makers “believed” issues for which there were no previous tasks (issues they or the usability engineer had not anticipated).
2. Fixing the discovered issues was planned for and agreed upon prior to testing.
3. The usability feedback was delivered as soon as it possibly could be –right after the issues occurred.
4. The team had measurable assurance that the solutions were successfully fixing the problems because the fixes were tested by the subsequent participants. In addition the team caught “poor” fixes for problems and corrected them.

It is important to reiterate the reasons that the RITE method succeeded for the Age of Empires II tutorial. They were:

- The attendance of product decision makers at the tests (developers, usability engineer, program manager, etc.).
- The rapid identification of issues and potential solutions.
- The identification of tasks that users must be able to do.
- The usability engineer had a great deal of experience watching participants in similar situations in the past, and was very versed in the product itself.
- The developers had a strong knowledge of design and a deep understanding of the system architecture.
- The opportunity to brainstorm different fixes as testing occurred.
- There was agreement between decision makers on changes to be made.
- Age of Empires II had a powerful and flexible architecture of the scenario creation editor, which allowed for rapid changes to be made to the product.

As stated in the introduction, the RITE method is not “new”—practitioners do this (and other activities like it) all the time. But there is surprisingly little said about methods like it in the literature. The practice of making changes after having run 1-3 participants appears to be common. A quick internal survey of usability engineers and practitioners at Microsoft and on a public list service (Utest) found that 33 of 39 respondents had used a similar method of very rapid iterations and fixes at least once. Lewis [11] recommends essentially running 3 participant usability studies with fixes in between. In addition, Nielsen [13] has pointed out that given the option it is better to run more tests with fewer participants because that is the most efficient method for both uncovering problems and verifying fixes. The RITE method differs from Lewis and Nielsen’s recommendations in that changes do not occur after a set number of participants, and verification of those changes are planned during the course of testing.

If usability engineers are considering using the RITE method for themselves they should first consider whether they have the right mix of “ingredients” similar to those listed above. In addition there are dangers to using the RITE method. Some of them are:

- Making changes when the issue and/or the solution to the issue are unclear. Poorly solved issues can “break” other parts of the user interface, or user experience. This happened a couple of times in the Age of Empires II test –although to the best of our knowledge we were able to catch these issues and fix them because of the structure of the RITE method.
- Making too many changes at once. If one of these changes degrades the user experience it may be difficult to assess which of the changes is causing the problem.
- Not following up at the end of the test with enough participants to assess the effectiveness of the changes made. Without this follow up there is no guarantee that the changes made were any more successful than the previously tested user interface.
- Missing less frequently occurring, yet important, usability issues. By using very small samples between iterations it is possible that less frequently seen issues will slip through unnoticed. This is particularly true if the task is broad and the domain is known to have many such issues [16], or the topic is one in which the researcher can not afford to miss an error.

When asked about the RITE method some usability practitioners we surveyed either wholeheartedly endorsed it, or reluctantly admitted to using it. The reluctant practitioners expressed the following reservations with the method:

- Reliability/Validity. The issues found by 1-3 participants might not be seen ever again. They might be abnormal or not the true phenomenon of interest.
- Power. With a small number of participants (1-3) they would not uncover all the issues that the user interface had.

With respect to reliability/validity, making a change based on minimal participants is a risk but one we are willing to take in certain situations. When the problem and solution are “obvious”, (as we would stipulate our wood gathering example was) – seeing an issue once is sufficient. In addition it is important to note that the vast majority of issues were not obvious and therefore did not get fixed until we had more clarity (e.g. seen more participants). A quick thought experiment should make our position clear. Consider the following two cases where the one participant in a usability test doesn’t seem to notice the difference between two color codes.

- Case 1: The color codes are Red and Green.
- Case 2: The color codes are Black and White.

In Case 1, if you know that Red-Green color blindness affects about 10% of males, and we discover through the verbal protocol that our participant is red-green color blind then it is clear that the red-green color states are a problem and it is clear how to fix the problem (don’t use both Red and Green, or add an additional cue like shape or size, etc.). In Case 2, it isn’t clear what is going on—there isn’t any such thing as Black-White color blindness (to our knowledge). If you confirm that the participant can see normally, then you’re bound to conclude this is random, inconclusive or simply bizarre. The key difference between the two cases is that the observation is evaluated against the knowledge and critical thinking ability of the observers—additional data aren’t needed in the Red-Green case, but they are in the Black-White case. In addition, in our view in a business context it is more important to establish the sufficiency of a solution rather than the reliability of observed problems. Thus, problems should be fixed as soon as they can be identified, agreed upon and a plausible solution proposed. These solutions can then be tested with subsequent participants to establish some confidence that they would work in practice.

With respect to power, we are far more concerned with the power needed to assure that a fix has actually solved a problem. Within the definition of the RITE method the only guideline is to run as many participants as is needed to be confident of the fixes your team has made. Having said this, it is absolutely correct that a small number of participants will uncover a small number of problems that are only likely to occur in a high frequency of the population. But this rests on the assumption that the researcher will only run 1-3 participants and that the interface for each batch of participants is entirely different. Our experience in using this method was that the likelihood of detecting problems actually increased by rapid iteration (note: figure 1 shows an increase in problems and failures in the early stages of the testing). This is due to the fact that much of the interface was still the same and as problems are fixed the participants completed more of the tasks without intervention and in effect “tested” more of the user interface.

Finally, Gray and Salzman [6] have argued that usability methods should be assessed using multiple techniques. We have applied that approach and although the results of each assessment (user response, and reviews) can be explained in other ways (and in fact are probably the product of multiple factors), our interpretation is that overall they provide substantial evidence in favor of the RITE method.

In conclusion, we believe that an understanding of the overall effectiveness of applied usability methods in different situations is still evolving. Previous work has established the relative effectiveness of different usability methods to efficiently uncover problems. However, some comparative studies have been criticized for their methodology [6]. We agree with Bonnie John [8] who suggests that an understanding of applied methodologies will develop through the careful reporting and discussion of a series of case studies of a variety of methods combined with clear and precise definitions of the method

used. We hope that this study helps foster such a tradition. We encourage others to replicate this work in other areas, find additional strengths and weaknesses and report them.

REFERENCES

1. Bias, R. G. and Mayhew, D.J. (1994). *Cost Justifying Usability*. Academic Press, New York.
2. Butler, M. B. and Erlich, K. (1994). Usability Engineering for Lotus 123 version 4. In M. Wiklund. (ed.) *Usability in Practice: How Companies Develop User Friendly Products*. Academic Press, New York, 1994. 293-327.
3. Card, S. K., Moran T.P. & Newell, A. (1983). *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum Associates, Hillsdale NJ.
4. Dumas J., and Redish J.C. (1993). A Practical Guide to Usability Testing. Ablex, Norwood, N.J.
5. Gray W.D. John, B.E. & Atwood, M.E. (1993) Project Ernestine: Validating a GOMS Analysis for Predicting and Explaining Real-World Task Performance. *Human Computer Interaction*, 237-309.
6. Gray, W. D. and Salzman, M.C. (1998) Damaged Merchandise? A review of Experiments that compare usability evaluation methods. *Human Computer Interaction*, 13 (3). 203-261
7. Gunn, C. (1995) An example of formal usability inspections at Hewlett-Packard Company. In *Proceedings of CHI '95 Conference Companion* (May 7-11, Denver, CO) ACM Press., 103-104.
8. John, B.E. (1998). A Case for Cases, *Human Computer Interaction*, 13 (3), 279-280.
9. Klemmer, E. T. (1989) *Impact in Ergonomics: Harness the Power of Human Factors in Your Business*. Ablex, Norwood, N.J..
10. Lewis, J. R. 1990. Sample sizes of observational usability studies: Tables based on the binomial probability formula. *Tech. Report 54.571*. Boca Raton, FL: International Business Machines, Inc.
11. Lewis, J. R. 1991. Legitimate use of small samples in usability studies: three examples. *Tech. Report 54.594*. Boca Raton, FL: International Business Machines, Inc.
12. Lewis, J. R. 1993. Sample Sizes for Usability Studies: Additional Considerations. *Tech. Report 54.711*. Boca Raton, FL: International Business Machines, Inc.
13. Nielsen, J., and Landauer, T. K. (1994). A mathematical model of the finding of usability problems," *Proceedings of ACM INTERCHI'93 Conference*. Amsterdam, The Netherlands, 24-29 April, 1993. 206-213.
14. Nielsen, J., and Mack. R. (1994). *Usability Inspection Methods*. New York: John Wiley and Sons.
15. Sawyer, P., Flanders, A., and Wixon, D. Making a Difference – The Impact of Inspections. In *Proceedings of CHI' 96*, (Vancouver, B.C., April 1996) ACM Press, 376-382.
16. Spool, J. and Schroeder, W. "Testing Websites : Five Users is Nowhere Near Enough. In Proc. CHI 2001, Extended Abstracts, ACM 285-286
17. Virizi, R. A. (1990). Streamlining the design process: running fewer subjects. In Proceedings of the Human Factors Society 34th Annual Meeting (Orlando, FL.) Human Factors Society, 291-294.
18. Virizi, R. A. (1992). Refining the test phase of usability evaluation: How many subjects is enough? *Human Factors*, 34, 457-468.
19. Wixon, D. R. and Jones, S. (1996). Usability for Fun and Profit: A Case Study of the Design of DEC Rally Version 2. In. M. Rudisill, C. Lewis, P. Polson, T. McKay (eds.) *Human Computer Interface Design: Success Stories, Emerging Methods, and Real-World Context*. Morgan Kaufman, New York, 3-35.
20. Woolrych, A. and Cockton, G., "Why and When Five Test Users aren't Enough," in *Proceedings of IHM-HCI 2001 Conference: Volume 2*, eds. J. Vanderdonckt, A. Blandford, and A. Derycke, Cépadès Éditions: Toulouse, 105-108, 2001

Usability inspection of digital libraries: a case study

H. Rex Hartson, Priya Shivakumar, Manuel A. Pérez-Quiñones

Department of Computer Science – 0106, Virginia Tech, Blacksburg, VA 24061, USA;
E-mail: hartson@vt.edu, pshivaku@vt.edu, perez@cs.vt.edu

Published online: 23 July 2004 – © Springer-Verlag 2004

Those that know the wheel of seasons

Notice nothing but its turning.

Time, they say, is love's dimension;

Love, the fruit of trial and tears.

The Fruits of Time, ROBERT K. FRANCE

Abstract. This paper reports a case study about lessons learned and usability issues encountered in a usability inspection of a digital library system called the Networked Computer Science Technical Reference Library (NCSTRL). Using a co-discovery technique with a team of three expert usability inspectors (the authors), we performed a usability inspection driven by a broad set of anticipated user tasks. We found many good design features in NCSTRL, but the primary result of a usability inspection is a list of usability problems as candidates for fixing. The resulting problems are organized by usability problem type and by system functionality, with emphasis on the details of problems specific to digital library functions. The resulting usability problem list was used to illustrate a cost/importance analysis technique that trades off importance to fix against cost to fix. The problems are sorted by the ratio of importance to cost, producing a priority ranking for resolution.

Keywords: Usability – Inspection – Digital library – Usability problems – Cost-importance analysis

1 Introduction and background

As part of our usability engineering role in the digital library project entitled *Collaborative Project: Core Integration for the National SMETE Digital Library* at Virginia Tech, we conducted a usability inspection of a digital library system called the Networked Computer

Science Technical Reference Library (NCSTRL), to be found at: <http://www.ncstrl.org/>. As our charge was to perform a formative usability inspection, this paper reports a case study about lessons learned and usability issues encountered, and not a formal lab-based usability test or a summative study involving user performance measures and statistics. In the course of our work we have also performed a usability inspection of iLumina, found at <http://www.ilumina-project.org/>, a digital library of shareable undergraduate teaching materials for science, mathematics, technology, and engineering where the user can contribute, locate, and download resources. We also have looked at the usability characteristics of several other digital library systems. We have observed that many of the usability issues discovered in our inspection of NCSTRL are generally applicable to all the digital library systems that we have explored. Thus, while our report is worded specifically in terms of NCSTRL, many of the concepts apply to other digital library systems as well.

An important category of common problems that we observed involves an apparently functionally oriented design approach rather than an approach based on user task threads. Another category involves the way that most of these digital library systems handle the submission of documents and the corresponding concept of metadata (descriptive data formatted for searching). Finally, many of these systems share problems with the use of words (e.g. in labels), including a tendency to use jargon and designer-centered, rather than user-centered, terminology.

1.1 National SMETE Digital Library (NSDL)

The National Science Foundation is developing the National SMETE (science, mathematics, engineering, and technology) Digital Library (NSDL, <http://www.nsdl.org/>).

The NSDL has a goal to create a community of users and information providers to enhance education at every level, including K-12, undergraduate, graduate, and life-long learning. The NSDL will be a repository of information and a central place for collaboration in improving education, primarily in the USA, but indirectly around the world. It will offer portals and collections to support various user communities (K-12, undergraduates, professors, computer science, different branches of engineering, sciences, etc.).

1.2 Our role in NSDL evaluation activities

Our area of work is human-computer interaction (HCI), especially usability engineering. In particular, we concentrate on formative evaluation in the usability engineering process [16].

All existing usability development methodologies agree that, to produce a product with high usability, a complete usability engineering process must be an integral part of the overall software-development process from the very beginning of the project. Typically, a usability engineering process involves client and user interviews, task analysis, user class definitions, usage scenarios, iterative usability design, prototyping, design walkthroughs, and usability evaluation. Unfortunately, many software developers think of usability engineering as only usability testing, to be done at the end of a project. Additionally, the reality of software-development cycles is such that time constraints, ill-specified requirements, and lack of usability-trained professionals often conspire to make usability issues an after-the-fact consideration. Thus, usability specialists are often called in at or near the end of the project to 'do some usability testing'. They are asked to help fix a user interface for which they had not participated in the design. They typically will identify problems that, had they been part of the development team, could have been addressed much earlier and much more economically.

The previous paragraph sums up our experience as practitioners representing usability in this project. We were not part of the design team that built NCCTRL, nor were we privy to the design discussions and design rationale that led to what we saw in our inspection. Although it would have been better to maintain close communication with the designers and developers while doing the inspection, this was not a practical option in our case. We were asked to conduct a usability evaluation after the system was deployed, and funding for further NCCTRL user interface work had expired. At the time of this inspection we hoped for additional funding, but it did not materialize. Currently, there is still some development being done on NCCTRL back-end functionality and we contacted a former developer for cost information for this paper but, in the main, we did our inspection more or less independently, without much interaction with the designers or developers.

These circumstances led to some disadvantages compared to the ideal situation but they, unfortunately, are not atypical:

- We, the inspectors, are usability people and not digital library people and that means that we are not intimately familiar with digital library conventions, designs, and issues. It is not unusual for a usability team unfamiliar with the application domain to be called in to do a usability evaluation.
- The inspection was not grounded in a deep understanding of NCCTRL's capabilities. Unfortunately this, too, happens frequently.
- The inspection did not ease communication between developers and usability people.

Despite these negatives, we feel there is value in reporting what we did, because it helps raise awareness of usability issues in the digital library domain. It also demonstrates that, even in difficult and highly constrained conditions, a usability inspection can yield useful information about usability defects. Finally, having done informal inspections of two other digital library systems, we think that NCCTRL inspection results are typical and representative of many different digital library systems and therefore valuable to designers, developers, and users of other digital library systems. We hope that, by reporting this work, we can contribute to the inclusion of usability practitioners in future digital library projects from the inception.

1.3 Usability inspection

Lab-based usability testing is often the method of choice for thorough formative usability evaluation. A lab-based formative evaluation process would involve employing real and representative users performing benchmark tasks, collecting qualitative and quantitative data to aid redesign. After deployment, remote usability evaluation methods can be used to continue gathering formative usability data from real users doing real tasks in their daily work. We are currently developing a method for remote users to report their own critical incidents [14] and are adapting it to digital libraries.

Our limited resources did not allow for full formative usability testing, leading us to opt for the less expensive usability inspection. Many real-world teams must make this same choice for the sake of cost effectiveness. Usability inspection [29] is a usability evaluation method that:

- Applies to early designs (e.g. low-fidelity prototypes), well-developed designs (e.g. high-fidelity prototypes), and deployed systems.
- Is usually less expensive than lab-based usability testing.
- Does not employ real users (in contrast to lab-based testing).
- Is expert-based (conducted by trained usability engineering practitioners).

- Is often, but not always, guided by user tasks.
- Has the goal of predicting usability problems that users will encounter in real usage.

1.4 Networked Computer Science Technical Reference Library (NCSTRL)

The Networked Computer Science Technical Reference Library (NCSTRL) [4] is a collaborative project involving NASA Langley, Old Dominion University, University of Virginia, and Virginia Tech. NCSTRL is a distributed system that provides a single point of access to the technical reports from participating international computer science departments and laboratories, through its Web interface at <http://www.ncstrl.org/>. Technical reports are gathered in NCSTRL via a process called ‘harvesting’ from other repositories. Harvesting is the process of issuing a request through a particular protocol to a data provider (a partner repository or archive) and using the returned metadata (data such as title, authors, date, etc., representing the document) as a basis for building value-added services, such as search engines, browsing capabilities, and output formatting. Submissions to NCSTRL are done indirectly through the Computing Research Repository (CoRR), an online repository of a partnership of ACM (Association for Computing Machinery), the Los Alamos e-Print archive, and NCSTRL.

2 Related work

Usability in general has seen enormous coverage in the literature; the book on usability engineering by Nielsen [28] and the usability engineering process books by Hix and Hartson [17] and by Mayhew [26] are representative. Digital libraries represent an important application area for usability and user experience, but in the digital library literature a focus on usability is only beginning to emerge and has lagged the non-user-oriented technical topics. As Dillon [5] puts it, “From an HCI perspective, much of the literature on DL research is highly technocentric.”. Dillon goes on to state that only one-third of the papers published in the 2000 ACM Digital Library Conference include user studies. The NSDL literature is similar, with significant efforts being devoted to creating collections [2, 8, 9, 18, 19, 23], protocols for sharing data [2, 3, 20, 22, 30], security issues [6, 11], and even user interface software infrastructure [11, 27, 35], but less to the usability aspects of the work [8].

One example of a brief evaluation of searching and browsing in NCSTRL, conducted by Theng [32], was aimed at exploring the problem of being ‘lost in hyperspace’ in the context of digital libraries. Theng evaluated searching and browsing by 10 participants, seven of whom were experienced users of digital libraries. The author reported that 30% of users experienced feeling lost, 40% could not identify where they were, and 80%

could not return to previously visited information. To address these problems, the author recommends that developers must “take into consideration both users’ and designers’ goals”. The paper does not recommend any more specific design changes to the NCSTRL interface to address the problems found in the study.

Another representative digital library usability report in the literature discusses the usability evaluation of Cypress, a digital library with approximately 13 000 color images and associated metadata from the Film Library of the California Department of Water Resources (DWR), a division of the California State Resources Agency [34]. The authors stress the importance of user-centered design applied to digital libraries, illustrating with issues and problems specific to the Cypress system. They grouped usability problems found (via user-based usability evaluation) by related usability principles.

In our work, we found that grouping usability problems by principles did not provide any new insights into the problems found. In our evaluation, we have grouped the problems by usability problem type and/or by system functionality. This method has helped us see clusters of problems and draw some important conclusions, which are described in Sect. 4.

The Cypress study also emphasizes user needs assessment, with which we strongly agree. Their iterative design process gives substantial up-front attention to the needs of users in the areas of content, resource discovery and retrieval methods, document analysis, interface design, and browsing.

Another usability report on digital libraries [33] is a preliminary report of the User Needs Assessment and Evaluation for the UC Berkeley Electronic Environmental Library Project, which is aimed at developing a massive, distributed, electronic, work-centered library of environmental information containing text, images, maps, sound, full-motion videos, numeric datasets, and hypertext multimedia composite documents to support actual environmental planning decisions. These researchers discuss the user needs assessment and evaluation components of the project: the underlying premises, methods, and initial findings.

A more formative approach, based on user studies, taken by the researchers at the Harvard–MIT Data Center (HMDC) at Harvard University and the University of Michigan’s School of Information and College of Engineering, is presented in the paper entitled ‘Usability Testing of the Virtual Data Center’ published in the Joint Conference on Digital libraries (JCDL) in 2002. The Virtual Data Center (VDC) is an open-source, Web-based digital library for the management and dissemination of social science research data.

Usability testing and heuristic inspections were an integral part of the development process from the inception of the VDC project and throughout its evolution. Among other issues, the paper stresses the negative effects on usability that result from a presupposition that users have

high levels of technical expertise. We strongly agree with this and have found many such problems related to the use of technical terms and specialized language in digital libraries that are not user centered. We discuss these in later sections of this paper.

Interested readers can find an overview of the history of NCSTRL in Davis and Lagoze [4] and of performance evaluations for NCSTRL server activities in Powell and French [31].

3 Method

3.1 Inspection instance

We performed the NCSTRL usability inspection in April 2002. It took about 24 person-hours for the inspection and about 16 person-hours to complete the original (online) report [16]. For the inspection, we used Internet Explorer version 5.50.4134.0600 on Microsoft Windows 2000. The evaluation was conducted on an AT/AT compatible Gateway computer with 130.6 MB RAM, using a 10-Mbps connection to the Internet. Our evaluation was conducted on NCSTRL as it existed around the time frame mentioned above, and we have not tracked changes or improvements made to the system since the inspection. We tried to preserve a copy of the evaluated version of NCSTRL for reference purposes. However, as it was not practical to establish a static version of the entire, database-intensive NCSTRL system merely as a reference for this report, we are unable to provide a persistent URL to the evaluated version of NCSTRL.

3.2 Pre-inspection preparation

In preparation for the inspection, we compiled a list of generic and representative NCSTRL tasks, but not benchmark tasks [17] or detailed scenarios, to represent the breadth of usage possibilities. We browsed and explored the system to become familiar with the overall look and feel and range of functionality.

3.3 Co-discovery technique

The primary method used for this usability inspection is sometimes called co-discovery [12] in which two or more evaluators work together. Co-discovery often leads to rich verbal protocol arising from the conversations among the evaluators. In our study we used three evaluators (the authors) with different characteristics, in order to get a broad representation of user perspectives. One evaluator was completely new to NCSTRL; another evaluator had enough experience with NCSTRL to know what kinds of tasks are appropriate; and the third had been working more extensively with digital libraries in general. Working in twos, or all three together, this variety in backgrounds allowed us to understand design constraints

and conventions behind some usability issues while not overlooking impressions that a new user might form. Furthermore, one of the evaluators is involved in the design of CITIDEL and was, at the time of the work here reported, involved in design decisions that would produce a system similar in functionality to NCSTRL. This provided the team with extra expertise and understanding from an implementation point of view. This is a perspective that is often ignored in usability reports, which tend to focus only on the user experience. Having an understanding of the design decisions made or considered to reach a particular user interface implementation allowed us to provide better feedback on what possible approaches could make the user experience better.

Despite our range of digital library background knowledge, none of us works in digital libraries as a primary field of interest. This lack of depth in our experience with digital libraries was both a benefit and a disadvantage. Our relative naiveté with respect to digital libraries allowed us to operate without the biases of digital library tradition and conventions (of which, we learned, there are many) and perhaps to consider less constrained design ideas. In other cases, we could provide an independent confirmation of larger technical issues, such as those for combining search and browse functions, broadly discussed in the digital library community but of which we were not completely aware before the inspection. On the other hand, in some cases we could have used more knowledge about digital libraries. For example, it would have been useful to know when digital library conventions were strong enough to color user expectations to the point where they might override our perceived usability issues.

3.4 User classes

The primary user class for NCSTRL is composed of scientific researchers in the field of computer science, who use NCSTRL to search and browse technical reports, retrieve documents, and submit their own technical reports via CoRR (Computing Research Repository), from which the reports are harvested by NCSTRL. This user class was the only one represented in our usability inspection.

A group of ‘administrators’ also uses NCSTRL, to keep the system updated by harvesting metadata on a regular basis from the various sources that support the Open Archives Initiative. However, the administrators use a command-line interface and not the usual Web-based user interface that we inspected.

3.5 User tasks

Our approach to usability inspection was task-driven and we used the following high-level categories of representative tasks to organize our inspection process:

- **T1 – Search** for technical reports based on a set of criteria.

T1a – Simple Search involves searching all bibliographic fields, grouping the results by archive, and sorting.

T1b – Advanced Search provides for searching on specific bibliographic fields with several filter options.

- **T2 – Browse** the NCSTRL collection.
- **T3 – Register** with NCSTRL to publicize (advertise) reports through NCSTRL.
- **T4 – Submit** a technical report to CoRR.
- **T5 – Harvest** from NCSTRL using the Open Archives Initiative protocol suite.

Because of the specialized task domain requirements of task categories T4 and T5, our inspection involving these categories was limited and we recommend further usability testing with real users doing real document submission and harvesting tasks.

4 Results and discussion

As is usually the case for a usability inspection, or any method for formative usability evaluation, the results were in the form of a list of usability problems, each problem a candidate to be fixed in an effort to improve usability in the interaction design. In our presentation of these problems, we have grouped them by usability problem types (such as wording problems and consistency problems) and/or by system functionality (such as search function and browse function), rather than by severity (until the cost-importance analysis in Sect. 5).

Wherever usability problem numbers (e.g. Overall.1 and T1a.2) appear (e.g. in the cost-importance analysis spreadsheets shown as tables in Sect. 5), the problem numbers indicate the related task per the task labeling given in Sect. 3.5. These are also the same problem numbers used in the sample-evaluation report [16].

4.1 Good points about the NCSTRL design

The purpose of usability inspection is to find problems. Thus, a list of usability problems at the end of the day is an indication of success for the process and should not spell embarrassment or criticism for the designers themselves. However, because a list of problems can often be perceived as only negative and critical, especially by developers who have an ‘ownership’ stake in the design, usability evaluators wishing to avoid being perceived as the ‘usability police’ should not overlook the good points. To this end, we must say that we did find NCSTRL to be generally easy to understand and not difficult to use. The system functionality seems appropriate and the user interface is aesthetically pleasing. In fact, we found no ‘show-stopper’ usability problems that must be fixed, regardless of resources available.

The goal of a usability report is, however, to list what the evaluators perceive to be usability problems, and that is the thrust of this paper. Although the discussion of the

problems here is specific to NCSTRL, since we have found that many of the problems reported here are common to numerous digital library systems we feel that this discussion will also be useful in a more general context of digital library interaction design and evaluation. It is our hope that this report will help raise usability awareness, leading to a clearer understanding of user experience issues among digital library system designers.

4.2 Organization of usability problems for reporting

The raw output of a usability evaluation (lab-based or usability inspection) is an unorganized ‘laundry list’ of usability problem descriptions. Our definition of a usability problem is anything that impacts the user’s task performance or satisfaction. This takes the concept beyond the popular misconception that usability problems are only about look and feel, to problems for the user relating to system functionality and its usefulness [24].

We carried out the analysis by organizing the problems around usability issues and then using the problem descriptions to illustrate and relate to these issues. We began by printing out each usability problem description on a separate piece of paper. We then labeled each one with one or more ‘keywords’ that we thought were the best descriptors of the problem type, and sorted them by type. We found that some of these categories were what we called ‘the usual kind’, problems commonly found in most applications (especially GUIs) and not specific to a particular NCSTRL function. Other problems were very closely tied to the design for a digital library function. We further sorted the former class into problems about precise wording, consistency, graphic layout, and organization, and the user’s model of the system. The latter class was primarily about browsing, filtering, searching, and document-submission functions.

This simple sorting provided a modest chance at manual data visualization and confirmed to us the value of data visualization for usability for showing the broader picture of relationships among the data. Despite doing the evaluation, writing up the separate problem reports, and organizing and posting the whole thing on our project Web site, we were never completely aware of the nature of the data we had collected. Perhaps this was because, in such projects, the evaluators often find and report each problem in isolation.

In any case, we were surprised to discover that problems about wording, not directly associated with a particular function, accounted for 10 of our 28 total problems, or almost 36%. To someone not familiar with cognitive affordances [13] in interaction design, problems with wording may seem unimportant because they are ‘just about words’. However, precise use of words in user interfaces is one of the utmost important design considerations for usability. Clear, complete, and correct wording of button and tab labels, menu choices, and Web links is crucial to helping users, especially new users, learn and

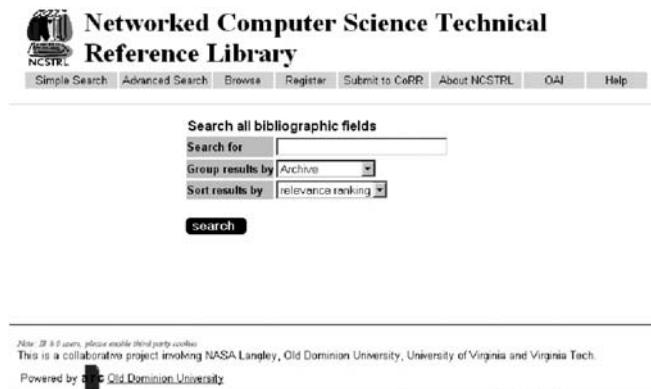


Fig. 1. The NCSTRL user interface

understand the functionality behind those interface controls. Additionally, since fixing wording often yields the greatest improvement in usability for the least cost, problems about wording are among the most important kinds of usability problems to solve. NCSTRL's high incidence rate of wording problems pointed out the value of having a person with strong writing and word skills on the interaction design team.

Another realization, perhaps less surprising but interesting, was that we had five problems (almost 18%) concerning the search and browse functions. This helped us look at those problems together and realize that the interaction design (and possibly underlying functionality) for these functions should be reconsidered together and would probably benefit significantly from a completely new, more integrated, design. The manual data-visualization exercise helped us see this need and the redesign requires a level of integration developers cannot get from looking at the problem descriptions one at a time. Furthermore, from an implementation perspective, searching and browsing are usually thought of as two separate functionalities. But, from a user perspective, they are so closely related to a single task (e.g. finding a resource) that they should be designed together.

4.3 The usual suspects

Finding usability problems of the 'usual kind' indicates that digital libraries are susceptible to the same kinds of problems as are many other applications. As context for the various problem descriptions, Fig. 1 shows the NCSTRL user interface and its three frames, a small one at the top for the tabbed menu of functions, a small one at the bottom for information on the project collaborators, and the large one in the middle for the main interaction. The top and the bottom frames remain constant throughout the interaction.

4.3.1 Problems with consistency

The terms **Group** and **Archives** seem to have been used interchangeably (Overall.5), both referring to institutions

SEARCH RESULTS GROUPED BY ARCHIVE		
ARCHIVE : AI Group at NASA		
Groups		
archive	Hits	
AI Group at NASA	2	
Auburn-Eng	86	
BostonUniv	189	

Fig. 2. Inconsistent use of terms **Groups** and **Archive** referring to institution

and to collections belonging to institutions. (Note that wording quoted from the NCSTRL user interface is distinguished by boldface font.) For example, the middle of the three search parameters in Fig. 1 shows **Group results by Archive**. At the left in Fig. 2 the term **Groups** is used to refer to an institution with a collection of reports. At the right, however, the term **ARCHIVE** is used instead. Two or more terms referring to the same concept can be confusing to the user and can impair learning. This kind of inconsistency can at least have a mild impact on new and returning users, who will often have to take the time out from a task to look for evidence to confirm that these terms are, in fact, used in the same way. This kind of terminology variation occurs naturally in a project before the terminology is 'standardized', but designers must root it out, to avoid confusion to the users. Both consistency and precision might be gained by using the term 'institution', for example, in both places. The final choice for a consistent term (as for any terminology question) is, of course, up to the users, who are subject-matter experts.

Another design point (T1a.4) about consistency relates to the difference between the **Simple Search** tab (far left in the top tab bar in Fig. 1) and the main label for the simple search function (at the top of the middle frame in Fig. 1), which is **Search all bibliographic fields**.

Usability designers view this technically as an inconsistency because the label at the 'point of departure', the **Simple Search** tab, and the label at the 'destination point', **Search all bibliographic fields**, are not close enough as a match for all users to be sure they arrived at the right place. It is easy to make a better match, boosting user confidence that they have arrived in the correct place, by saying 'Simple Search: Search all bibliographic fields', as shown in Fig. 3.

4.3.2 Problems with feedback

Feedback by highlighting the currently selected task or function is fundamental for providing a 'You are here' style of navigational grounding for the user. Unfortunately, clicking on a 'tab' (a menu button in the top frame of Fig. 1) does not result in highlighting or any kind of feedback as an indication of which tab is the currently active choice (Overall.2). This is typical of HTML designs that do not include some form of server processing to detect which page is being served and disable the link or change the color of the button (see Sect. 4.3.4 below for other problems with this navigational bar).

Fig. 3. Proposed matching of departure tab and arrival screen title

The design also has an instance of non-persistent highlighting as feedback (T1a.10). On the search-results page, the institution (archive) selected in the left-hand frame is highlighted when clicked, indicating the source of the contents displayed in the right-hand frame. However, the highlighting is not persistent and disappears after a few subsequent user actions. The solution is to maintain the link highlighting until another link in that group or at a higher level is clicked. In general, feedback used as long-term state information should persist beyond the immediate user action that sets it.

4.3.3 Problems with wording

In several places within the design, wording for labeling (especially when used as cognitive affordances for new users) uses jargon or slang or is unclear or missing. In some cases, where software code is apparently being reused, the wording is even incorrect.

Unclear, imprecise, and incomplete wording. A good example of terminology that was unclear to us, as new users, and that appeared to be jargon, or at least did not seem to us to be user-centered, is the term **Discovery Date** (Overall.3 and T1b.3). This term is used in several places in the interface; one example is the second choice in the pull-down menu **Sort Results By** for the simple search function, as shown in Fig. 4.

We were told by an NCSTRL developer that **Discovery Date** is the date that the author (or contributor) creates the NCSTRL record representing the document. This is not necessarily the same as the date on which that record is actually entered into the digital library of NCSTRL, which is called the accession date. The difference is due to the harvesting process that occurs between record creation and accession into NCSTRL. Use of the word ‘discovery’ in the user interface is enigmatic, as the usual concept of discovery does not seem to apply. It is probably a developer’s term showing through to the user, and a more user-centered term, such as ‘record creation date’, is indicated. Designers might have to survey some users to determine what term the average user is likely to use here. Furthermore, it is not clear to us why users would care when the record was created. It seems more likely that users will search for the publication date and not the date a record was created in a database. This information

might be used for administrative and/or record-keeping purposes, but would then play a role only with administrators and not with scientific researchers as users (see Sect. 3.4 for the user classifications).

Another way wording can be unclear and imprecise to new users is by being incomplete. An example is the **Submit to CoRR** tab at the top that does not say what is to be submitted and would be more complete if it said ‘Submit Technical Report(s) to CoRR’ (T4.1). Another example is the label, **Search all bibliographic fields**, in the simple search that does not say what or where the user is searching and would be more complete if it said, including the suggestion made earlier, ‘Simple Search: Search all bibliographic fields in selected archive’ (T1a.6) or, per the discussion in Sect. 4.3.1, ‘Simple Search: Search all bibliographic fields for selected institution’. Additionally, there is plenty of space also to add a sentence about what the term ‘archive’ refers to in this context – for example, ‘Each archive contains the technical reports from one institution’. Although long labels can be unwieldy, they are sometimes necessary for increased completeness and clarity for the user.

This added wording might also be part of the solution for the problem (Overall.5) of using the terms **Group** and **Archives** referring to collections belonging to institutions, discussed earlier in Sect. 4.3.1 and seen in, for example, the heading of the left-hand column at the top of the left frame of the browse page, as shown in Fig. 2.

Another example (T1b.3) of wording that should be made more complete is shown in Fig. 5, where the terms **Archive’s Set**, **DateStamp**, and **Discovery Date** (discussed above) are anything but self-explanatory and obvious in their meaning to users.

All these terms seem to be designer-centered terms that need to be changed to user-centered terms and briefly explained. The term **Archive’s Set** is a technical term and stands for the ‘sets’ concept in OAI-PMH (The Open Archives Initiative Protocol for Metadata Harvesting), which is not user-centered. In OAI-PMH, a set is an optional construct for grouping items for the purpose of selective harvesting.

One could argue that users who are doing the tasks involving these terms will know what the terms mean. However, making the words more complete and precise makes a better fit to the tasks and will give new users faster recognition as a match to a given task. Even if new

Fig. 4. Use of the term **Discovery Date** in a simple search menu

The screenshot shows the 'Advanced search' section of the NCSTRL interface. It includes fields for 'Author', 'Title', and 'Abstract', each with a text input field and a 'search' button. Below these are 'Filter options' for 'Archive' (set to 'All'), 'Archive's Set' (set to 'All'), 'DateStamp' (text input), 'Discovery Date' (text input), and 'Display options' for 'Group by' (set to 'Archive') and 'Sort by' (set to 'relevance ranking'). There is also a general 'search' button at the bottom.

Fig. 5. Advanced search, as seen in NCSTRL

users are likely to make the right choices eventually, they might first have to look for and eliminate other possibilities to be sure that this is the best match to the task, incurring an unnecessary productivity penalty.

Incorrect or inappropriate wording. On clicking the **Browse** tab and selecting an institution at the left (apparently any institution except the first one), the labels **SEARCH RESULTS GROUPED BY ARCHIVE** and **hits (1-n) of total xxx hits** displayed at the top of the right hand side (see Fig. 6) are incorrect (T2.1). The top label refers to the search function, even though we are in the browse function and the second label refers to the pages and reports as though they have been retrieved via a search when they really are just reports available for browsing. These words appear to come from code shared with the search design. The words should say ‘Browse reports, grouped by institution’ and ‘This is page 1 of a total of 42 pages for the selected institution, reports 1–10 of a total of 418 reports’.

Finally, in several places in the NCSTRL interface, the term ‘hit’ or ‘hits’ is used to refer to documents retrieved, as seen for example in Fig. 6 (T2.4). While this is not an uncommon term in the world of information and retrieval, in the usability world it is considered inappropriate slang. At best, it is unattractive and, to some, it is slightly offensive. Although this is only an aesthetic or cosmetic consideration, we do recommend using something more descriptive, such as ‘Matches with search term’.

4.3.4 Problems with layout and graphic design

Many of the visual cues new users get from GUI and Web designs are from the appearance of the graphic design, especially from how things are located and grouped by related function on the screen. Because of the importance of visual design to user performance, software-development teams increasingly are adding graphic designers and ‘information designers’ to the group doing interaction design. The top menu bar of NCSTRL (Fig. 1)

The screenshot shows the 'SEARCH RESULTS GROUPED BY ARCHIVE' page. At the top, it says 'This is page 1 of total 9 pages, hits (1-10) of total 86 hits.' Below this is a 'Results Pages' navigation bar with links 1 through 9. The main content area shows a result for 'ARCHIVE :Auburn-Eng' with 'Title' 'Performance of LAMS-DLC, a Protocol for Networks' and 'Authors' 'Ward, Christopher | Choi, Cheong H.'

Fig. 6. Incorrect labeling for browse function

is a good example of a visual design that needs reorganization in terms of information content (Overall.7). Often menu items get added to a design in the order the functions are considered by designers, an order that might not mean much to the user. After the menu choices are established, designers can improve usability by reorganizing them and grouping by related task or functionality. Organizing tasks by major task categories can present a more structured system model and reduce cognitive workload for users. Choices in the NCSTRL menu (Fig. 1) lead to a mixture of both information-seeking tasks and other interactive functionality.

A natural abstraction of these tabs might be represented by groups labeled user tasks and information links. The user tasks would then contain Simple Search, Advanced Search, Browse, Register, and Submit Technical Report to CoRR. This last item has other problems of its own (T4.2) (see Sect. 4.3.2). The information links would contain the items About NCSTRL, About CoRR, OAI, and Help as shown in Fig. 7 below.

The menu tabs have a further graphical problem. The gray background to the links in the top menu bar makes them look like buttons, as shown in Fig. 1, but they do not behave like buttons (Overall.6). The user might click on the background surrounding a label, assuming it is a button, but not get any result. Because the ‘button’ is actually just a hyperlink, it requires clicking exactly on the text in order to get a result. We call this a false affordance, since it has a false visual appearance that elicits an erroneous user action. Real graphical buttons are relatively easy to use in place of these links.

The **Advanced Search** tab clearly supports an advanced search as a primary task. However, as another example of graphical design as a cognitive affordance for the user, designers should consider adding an Advanced Search button to support an advanced search as an extension of a simple search (T1a.5). When a user tries a simple search and finds it inadequate, the next obvious step is to try the advanced search. At that point of this task sequence, having an Advanced Search button at hand would support the user’s workflow or task thread continuity. It would be easy to include an Advanced Search button on the simple search page, as shown in Fig. 8. Although redundant with the **Advanced Search** tab at the top, this button will appear at the right time and place

User Tasks					Information Links			
Simple Search	Advanced Search	Browse	Register	Submit TR to CoRR	About NCSTRL	About CoRR	OAI	Help

Fig. 7. Proposed main menu, reorganized by user tasks and information links

Search all bibliographic fields	
Search for	<input type="text"/>
Group results by	Archive <input type="button" value="▼"/>
Sort results by	relevance ranking <input type="button" value="▼"/>
Search	

You can also perform an advanced search with more options. For that, click below.

Advanced Search

Fig. 8. Proposed addition of an Advanced Search button to support task thread continuity for the user extending the simple search

for a user who needs to extend a simple search into an advanced one.

Another graphical design issue is about how the proximity of elements in interaction design layouts is used to indicate associations and relatedness. As shown in the top part of Fig. 5, the **Search** button (for the top section of the advanced search) is located very close to the **OR** choice and in the line labeled **Combine fields with** (T1b.1).

However, the **Search** button is not intended to be associated directly with either the **OR** choice or the **Combine fields with** label. The **Search** button is intended to be associated equally with everything in the **Search specific bibliographic fields** box; i.e. the button goes with the entire search function or task. This broader association can be indicated by moving the button further away and up to the middle, as in Fig. 9.

Our final example of the need for graphic design to improve the layout to group user interface elements by related function is seen for the advanced search in Fig. 5 (T1b.4).

The pieces of the display that support the advanced search task should be organized and grouped together by related function and the **Search** button should be moved to show its association with the entire search function. Figure 10 shows a possible design solution.

Search specific bibliographic fields	
Author	<input type="text"/>
Title	<input type="text"/>
Abstract	<input type="text"/>
Combine fields with <input type="radio"/> AND <input type="radio"/> OR	
search	

Fig. 9. Proposed improved layout location of search button to associate it with the entire search function

Advanced Search	
Search specific bibliographic fields	
Author	<input type="text"/>
Title	<input type="text"/>
Abstract	<input type="text"/>
Combine fields with <input type="radio"/> AND <input type="radio"/> OR	
Search with filters	
Institution	Virginia Tech <input type="button" value="▼"/>
Archive's Set	All <input type="button" value="▼"/>
Use display options	
Group By	<input type="button" value="Institution"/>
Sort By	<input type="button" value="Relevance Ranking"/>
Search	

Fig. 10. Possible design solution for grouping by function within advanced search layout

In this design solution, the user would immediately know that the search button is associated with the entire set of fields, whereas the original design implies two separate searches, one a plain bibliographic field search and the other with **Filter options** and **Display options**, which is not actually the case. Also, the indentation of the **Filter options** and **Display options** could cause confusion to the user. The **Filter options** are indented one step and the **Display options** are indented further within the **Filter options**, which does not convey any necessary meaning.

4.3.5 Problems with user's system model

New users often expect the home page of an application to be a kind of 'splash page' that introduces the application and tells users how they can use it, thereby helping the user with planning about how to use the system to achieve goals in the work domain. In contrast, the NCSTRL has no home page, as such; it uses the simple search screen as the default screen, as seen in Fig. 1, and in so doing makes simple search the hidden (implicit) default task or function for the user (Overall.1 and T1a.1).

New users, arriving at the simple search screen without clicking on anything, may wonder what that is and how they arrived at it. A design that starts the site off with a function that wasn't requested by the user presents a confusing model of the system and can seem to some users as though the system has taken control from the

user and presented the simple search function possibly as though the user must start with a search.

If the simple search could be justified as the default function upon entering the system, then the design should follow the usability principle of making default choices clear. The redesign solution could be as simple as highlighting the **Simple Search** tab initially. At least that would show that the simple search is selected upon arrival, explaining why users see what we see in Fig. 1 immediately upon arrival.

However, we recommend, instead, designing a separate home page for NCSTRL to help new users understand what NCSTRL is and how it can be used. Expert users can by-pass the home page with one click on the tab of their choice.

4.4 Problems more specific to digital library functions

Some of the usability problems in the results were more specific to digital libraries. These problems also tended to be more complex and often related to how some of the core services [21, 22] are presented to users. We are now aware that these problems are difficult and that the information-retrieval community has been struggling with some of them for years. We think that it is important, however, to show that an independent evaluation not biased by a detailed knowledge of these struggles has pointed out the same problems, confirming their importance as ongoing research issues.

4.4.1 Searching, filtering, and browsing

In our usability inspection, we found several aspects of the searching, filtering, and browsing functions to merit consideration for redesign. It is possible that a single integrated redesign could solve most of these problems at once, given significant collaboration with designers and other developers.

Searching, filtering, and browsing are core services to almost all digital libraries and portals. As Robert France put it [7], users see searching, filtering, and browsing just as ways of selecting and navigating through documents and document surrogates to find information or resources to meet a need in their work domain. In this perspective, search criteria and filters look pretty much the same. Each is a logical criterion (query) that in some sense defines a certain subset of the full collection that the user expects to be ‘interesting’. Each is performed indirectly by the user in that it is automated by the system on behalf of the user.

In contrast, browsing is performed directly by the user through explicit acts of navigation within information structures. During this navigation, the user manually decides which documents encountered are ‘interesting’. One of the most common types of browsing is navigating through lists of results retrieved by a search. In this case, one could say that the search was used to filter the

information structure and the browsing was used to navigate the filtered information structure. Another kind of browsing is navigation through a list (or other structure) of documents in a given category. Filtering can be applied to either searching or browsing. If a filter is applied to a search, the result to the user is usually the logical AND of the search criteria and the filter criteria. That result can be interpreted as the filter pruning the search space for the user. If applied to browsing, the filter prunes the browsing space for the user.

To developers, these issues are somewhat different. Searching and browsing seem to be provided as separate functions in many digital library systems, as they are in NCSTRL. We believe this is often because developers see these functions as having different underlying mechanisms and follow a functional, rather than a task-oriented, approach to interaction design. The difference between a filter and a search might be that search criteria include attributes for which developers can build effective indexes to speed internal searching. Filters, especially for post-search pruning, might contain attributes for which record-by-record matching is necessary. By following a functional view of keeping them separate (e.g. invoked by different tabs), the user may be discouraged from using them together in a more integrated way where the two functions can work together.

Combining searching, filtering, and browsing. The above discussion leads us to one design consideration we suggest that most broadly affects many different digital library interfaces: the consideration to combine the search, filter, and browse functions into a single, powerful data-selection and navigation capability. This approach has been suggested for other kinds of information systems and database systems [25] but, for digital libraries, this point is only conjectural and would have to be discussed at length with digital library designers and users, taking into account other factors such as tradition, conventions, user expectations, and other task threads that might require separate search, filter, and browse functions.

Although it is a different domain and a different kind of interaction, perhaps digital library designers can take a suggestion from the domain of Web searching. Increasingly Web users are demanding more flexibility in searching and more power to customize searches to match their own information needs [10].

Supporting iterative query refinement. As shown in Fig. 11, the display of query results does not also show the query that led to those results (T1a.2).

This search appears to be designed as a one-shot function where the user submits a query, gets the results, and is done. However, from empirical usability data, we know that when users submit queries, the results are not always what they expected. The next logical step for the user in this task thread is to compare the query and the results to see how the query should be iteratively modified to get the desired results. Going back and forth between

SEARCH RESULTS GROUPED BY ARCHIVE	
ARCHIVE :CNRI	
Title	<i>National Digital Library of Theses and Dissertations: A Scalable and Sustainable Approach to Unlock University Resources</i>
Authors	Fox, Edward A. Eaton, John L. McMillan, Gail Kipp, Neill A. Weiss, Laura Arce, Emilio Guyer, Scott
Archive	CNRI
Discovery Date	1996-09-15
Document ID	oai:ncstrlh:cnri_dlib:cnri_dlib//september96-fox

Fig. 11. Display of query results

the query and the results on different screens introduces a workload on the user's limited human memory capacity. Having the query shown along with the results supports direct iterative query refinement.

Further, users often wish to prune the retrieved set of documents by applying a subsequent query to the results themselves. The need for this capability is quite common, being essential to support a divide-and-conquer approach to information seeking. But NCSTRL does not directly support the ability to search again, using a different query, within the results of a previous query (T1a.8). A design that supports the continuity of this essential query refinement task threads will show the query along with the results and will allow applying a query to the results in addition to searching the entire archive.

A minor additional limitation is that NCSTRL allows browsing only for documents grouped by archive (institution) and does not allow browsing technical reports grouped by author, year of publication, or other attributes (T2.3). We do not know how important these tasks are to most digital library usage, but the user community of most digital library systems is very broad, requiring broad task support in the system.

4.4.2 Inter-system navigation: the 'portal pass-through' problem

According to the glossary found at <http://www.auburn.edu/helpdesk/glossary/portal.html>, a portal is an entry point or starting site for a portion of the World-Wide Web, combining a mixture of content and services and attempting to provide a personalized 'home base' for its users with features like customizable start pages, filterable e-mail, a wide variety of chat rooms and message boards, personalized news and sports headlines options, gaming channels, shopping capabilities, advanced search engines, and personal home-page construction kits. Most of these portals started simply as search engines, but now have added other services such as email, chat, etc., and have thus transformed themselves into Web portals to attract a large audience.

In the context of digital libraries a portal could be thought of as a single point of access to distributed systems that provides services to support user needs to search, browse, and contribute content, often linking to shared existing functionality at other sites.

The practical effect in NCSTRL (and many other digital library systems) is a single portal Web page through which each participating site or system is accessed. Just listing the participating sites and giving links to them on this portal page makes the page little more than a high-level index to the other sites. In order to present a more integrated view of functionality across the systems of the whole portal, sometimes specific functionality is mentioned in the portal, but only as a link to that functionality in the other sites. This is the source of the 'pass-through' usability problem, a large issue for NSDL and other distributed digital library facilities, and one that we were not able to resolve entirely. It is especially important with respect to the Integration Project, for which we have some responsibility.

Half-way link to Submit to CoRR. The specific case of the NCSTRL link to CoRR for technical report submission is a good example of the portal pass-through problem (T4.2). The NCSTRL link **Submit to CoRR** is indirect (a 'part-way-there' link), leading not directly to a CoRR page for submission, but to the home/welcome page of CoRR, which in turn has a link for submitting a technical report, embedded within its text, as shown in Fig. 12.

This indirection offers poor support for the user's task thread. Upon arrival at the CoRR home page via the **Submit to CoRR** NCSTRL link, the user is expecting immediate access to functionality for submitting a document. Once it is apparent that this is only the CoRR home page, the CoRR link for submission may still not be immediately obvious. The user who can eventually find it, however, is again deflected from the task at hand, because that link actually leads to a dialogue box for logging in. Nor does this terse dialogue box explain that logging in is required here or why; it just offers the possibility to log in and stands in the way of doing anything else – an unarticulated indication to the user that logging in is required before submitting. Of course, logging in requires a password, having a password requires being registered as a user, and so on, moving further and further from the submission task, walking the user backwards through a task sequence revealed only a step at a time while requiring users to stack the intervening unexpected tasks in their mental models. There is much information overloading occurring in the login dialogue box and it does not support the user's task thread well. Among the usability

The Computing Research Repository (CoRR)

Welcome to the Computing Research Repository (CoRR), sponsored by ACM, the arXiv.org e-Print archive, NCSTRL (Networked Computer Science Technical Reference Library), and AAI. From here you can

- [browse](#) the Repository and peruse recent submissions or browse the entire NCSTRL collection
- [search](#) the Repository or the NCSTRL collection
- [subscribe](#) to the Repository or the NCSTRL collection, and get regular email notification of new submissions
- [submit](#) a document to the Repository using web upload, if you have already registered
- [get help](#) on many topics, including
 - [submission](#), including registration and using email or ftp submissions (which do not require registration). (Click on [web upload help](#) in the submission help file to get more help on submitting by web upload and author registration.)
 - [searching](#)
 - [submitting Latex source](#)
 - [downloading papers](#)
 - [other frequently-asked questions](#)
- [find out more](#) about CoRR, including a list of frequently-asked questions
- go to [mirror sites](#), for potentially faster access to CoRR if you are not in North America.

Papers in CoRR are classified in two ways: by subject area from a list of subject listed below and by using the [1998 ACM Computing Classification System](#). The ACM classification scheme provides us with a relatively stable scheme that covers all of computer science. The subject areas are not mutually exclusive, nor do they (yet) provide complete coverage of the field. On the other hand, we hope that they better reflect the active areas of research in CS. We expect to add more subject areas and sub-divide current subject areas according to demand. Authors who cannot find an appropriate subject area should use subject area Other. We

Fig. 12. Partial view of CoRR home page, containing the link to submit a document

guidelines that apply to this case are: use precise labels, ensure that responses to user actions are predictable, and avoid function overloading.

The portal pass-through problem is a very difficult problem, the most significant part of which is a complex system model that derives from using a portal to combine multiple systems, each with a potentially different method of operation, but without any real integration. Additionally, the model is more or less hidden from users. Just as an example of the hidden relationship between NCSTRL and CoRR, once a user has submitted a technical report to CoRR, it does not show up in NCSTRL until it is later ‘harvested’ from CoRR. Thus, regardless of the rest of the solution, an up-front description of how it all works is essential, perhaps from a tab or button labeled ‘About submitting reports’.

Beyond this up-front explanation of the system model, a possible solution could employ a ‘Submit TR to CoRR’ tab taking the user to a new page *within NCSTRL* that has its own ‘Submit TR to CoRR’ button, that explains the situation, and that serves as a ‘base of operations’ for going back to register, logging in, and moving forward to submit. In particular, this page in NCSTRL should:

1. Explain that submitting a technical report is a function within CoRR, not NCSTRL.
2. Explain that clicking the ‘Submit TR to CoRR’ button here will take you to the CoRR home page, where you will be in a different system.
3. Explain that a TR submitted to CoRR will not show up in NCSTRL until later, after a ‘harvest’ from CoRR.
4. Explain that the CoRR page will open in a new window and the back button will not return you to NCSTRL.
5. Explain that you will then have to click on the ‘Submit’ link in the CoRR home page.

6. Explain the need to log in to CoRR, in order to submit.
7. Offer a button leading directly (in CoRR) to the dialogue box for logging in.
8. Explain the need to be registered/subscribed as a CoRR user in order to log in.
9. Offer a button to go directly (in CoRR) to register.
10. Offer a button labeled ‘Submitting Technical Reports to CoRR’, to be used when the pre-requisite conditions regarding logging in and registering are already satisfied, to go directly (in CoRR).

This solution helps the user with the complicated mental model and supporting the user’s task stream. The extra click and page it requires is more than compensated for, and is not costly to the user since this task will not occur frequently.

Interestingly, developers of NCSTRL encountered one other form of the portal pass-through problem in regard to the formatting of distributed search results [4]. When a search from one NCSTRL site found documents on another site, the reports were displayed using the local site’s look and feel, that is, the look and feel of the site at which the document was stored. Davis and Lagoze [4] reported that the NCSTRL software was updated to produce a more consistent look and feel across sites. We expect that NSDL will follow suit once the effect on users is clear.

5 Sample cost/importance analysis

After data collection, we had typical usability data in the form of usability problem reports, along with our severity ratings of each problem. We then contacted a developer for estimates of the cost to fix each of the problems, after which we could perform a sample cost-importance analysis. The importance (to fix) ratings and cost (to fix) estimates here are just representative, to illustrate the analysis process. Each digital library project will have its

own goals and criteria and would assign its own importance ratings and cost estimates, plus their own rationale for these figures.

The purpose of cost-importance analysis is to help manage the design-change process in a realistic project environment where budget and schedule are limited, as explained in Chapter 10 of Hix and Hartson [17]. Because most development teams cannot afford to fix all usability problems found, they need a method for prioritizing, to provide solutions that offer the most effect on increased usability per unit of cost.

Cost-importance analysis is done in a spreadsheet where, for each problem or group of related problems to be considered together, developers estimate the importance to fix and cost to fix, calculate ratios of importance to cost, and sort by priority ranking. By drawing a cut-off line just before where cumulative cost meets or exceeds available resources, developers can delimit the problems which realistically can be fixed. All usability problems below the cutoff line are saved to fix if resources permit or are tabled until the next design iteration or next product version.

5.1 Importance (to fix)

The rating of importance (to fix) a usability problem is an engineering estimate and usability engineering practitioners get better at making this kind of estimate with experience. Because the importance rating is an estimate, it is best to keep the basis simple. We have chosen a scale of five integer values, allowing for fractions or decimals by interpolation [15]. In addition, the highest importance rating is a non-integer value of importance = M, meaning *must fix*, regardless of cost. This rating is used when a usability problem is a show-stopper or affects primary mission-critical or safety-dependent tasks. The numeric importance ratings are:

- Importance = 5: the problem has a major impact on task performance (e.g. the user cannot complete a key task), is expected to occur frequently, causes costly errors, or is a major source of dissatisfaction. Major problems that cannot be overcome by user learning are included (e.g. lack of feedback for an operation).
- Importance = 3: the user can complete the task, but with difficulty (e.g. it caused confusion and required extra effort) or the problem was a source of user dissatisfaction. Includes problems that might otherwise be rated as 5 but can be overcome by user learning (e.g. which button to click to do a certain operation).
- Importance = 1: the problem did not impact task performance or dissatisfaction much (e.g. was an irritant or a cosmetic problem), but is still worth listing.

The numeric values in between (2 and 4) are for interpolation in cases where one of the other ratings needs to be down-graded or up-graded due to, for example, a high or

low frequency of occurrence, the number of user classes affected, or the amount of user effort to recover. For example, a problem that is seen to block task completion is given an initial rating of 5 but, since the situation will not arise frequently and the task is not critical, the rating is down-graded to 4.

5.2 Cost (to fix)

Developers next consider possible solutions for each problem and estimate the cost to solve each, in person-hours. This is a stage at which it is often useful to combine related problems to be considered for a single solution that solves the whole group. Additional training is often suggested here as a solution, but is seldom a good solution, since the cost must be repeatedly incurred, for each new user, whereas an interaction design change that obviates the need for this training has a cost paid only once.

Cost figures given to us by NCSTRL developers, for example, were as low as 'a couple of minutes' to as high as a month. To allow for some overhead in considering each new problem and to dodge the problem of guessing how many minutes an estimate such as 'several minutes' means, we adopted the approach of rounding all cost amounts under one person-hour up to a value of one. Where the designers' estimate was 'several hours', we went with a half-day (four person-hours). Similarly, we used four days (32 person-hours) for estimates of 'several days'. Table 1 shows the major problems with their importance ratings and cost estimates.

5.3 Priority rankings

Before computing priority rankings, developers move all problems rated as M (something we did not have in this evaluation) to the top of the priority list. Sometimes the cumulative cost of the M-rated problems exceeds available resources, causing difficult management decisions and precluding fixing any other problems.

Using the problem data of Table 1, we then computed the priority ratios as $(\text{importance}/\text{cost}) \times 1000$. A higher ratio means more impact for the cost of fixing the problem. The figure of 1000 is just a scaling factor that ensures most ratios will be easy-to-use integers. We then sorted the problems of Table 1 by descending priority ratios, obtaining ordinal priority rankings, but the result was an ad hoc mixture of various seemingly unrelated problems. By grouping the problems according to the categories discussed in Sect. 4.2 we obtained a more well-behaved analysis, as shown in the spreadsheet of Table 2.

Suppose, just as an example, that the development team has two-and-a-half people available for one week to fix the problems. That is 100 person-hours and will just about cover all but the last group of problems, as seen in the final column of Table 2, the cumulative cost. Unfortunately, in this case, because of their high cost, the

Table 1. Cost-importance ratings for usability problems

UP number	UP description	Importance (to fix)	Cost (to fix, in person-hrs)
Overall.1	Home-page design	2.0	36.0
Overall.2	Tabs need feedback	3.0	1.0
Overall.3	Jargon (discovery date)	3.0	1.0
Overall.4	Inappropriate term (hits), slang	1.0	1.0
Overall.5	Inconsistent terms	2.0	1.0
Overall.6	Links that look like buttons, but don't behave like buttons	1.0	2.0
Overall.7	Tabs need organization	3.0	6.0
Overall.8	Consider combining search and browse functions	5.0	120.0
T1a.1	Hidden default for function/task	3.0	1.0
T1a.2	Can't see query and result together	5.0	1.0
T1a.3	Hidden default for search criteria	2.0	1.0
T1a.4	Simple search label needs to match tab	1.0	1.0
T1a.5	Advanced button in simple search	2.0	1.0
T1a.6	Search label not clear	1.0	1.0
T1a.7	Need for search box entry	1.0	1.0
T1a.8	Missing functionality	5.0	120.0
T1a.9	Needs user control of result display	1.0	3.0
T1a.10	Non-persistent highlighting	3.0	1.0
T1b.1	Spacing of search button	2.0	1.0
T1b.2	Imprecise filter options label	2.0	1.0
T1b.3	Unclear wording	4.0	1.0
T1b.4	Page layout for advanced search needs grouping by function	4.0	1.0
T2.1	Incorrect wording – search results	1.0	2.0
T2.2	Incorrect wording – result pages	2.0	2.0
T2.3	Limited browsing functionality	4.0	160.0
T2.4	Inappropriate wording – number of 'hits'	4.0	1.0
T4.1	Incomplete wording in a label as a cognitive affordance	1.0	1.0
T4.2	Half-way link to Submit to CoRR	5.0	36.0

Table 2. Cost-importance ratings for groups of related usability problems

UP group	Avg importance (to fix)	Total cost (to fix, person-hrs)	Priority ratio	Priority rank	Cumulative cost (person-hrs)
Feedback	3.0	2.0	1500.0	1	2.0
Consistency	1.5	2.0	750.0	2	4.0
Layout, graphic design	2.4	11.0	218.2	3	15.0
Wording	2.0	12.0	166.7	4	27.0
Navigation	5.0	36.0	138.9	5	63.0
System model	2.3	38.0	61.4	6	101.0
Search, browse functionality	4.0	404.0	9.9	7	505.0

troublesome search and browse functions fall below this cutoff line and will not get addressed, even though they have relatively high importance. Nonetheless, developers can begin discussing the redesign, in case they have a chance to make it in the next version.

report on the specifics of a usability inspection of a particular system (NCCTRL), the results and accompanying design discussion apply well to most other digital library systems, as well as other systems featuring search and browse functions and concepts such as portals.

6 Conclusion

Digital library systems are a relatively new area of application for usability design and evaluation. Although we

7 Future work

As future work, we are developing a tool, the Usability Problem Inspector based on the User Action Frame-

work [1], specifically designed to support usability inspections, but this tool was not ready to use for this project. We were, however, able to draw on the concepts behind the Usability Problem Inspector tool and we believe those concepts helped us conduct a more comprehensive inspection than we otherwise would have done.

Someday, we hope to have the kind of usability problem diagnosis, usability database, and data-visualization tools that would have supported our small-scale evaluation and analysis activities in this project, and that we think would contribute enormously to evaluation projects with large numbers of usability problem descriptions.

Acknowledgements. We give our thanks to the editor and reviewers for the very thorough and helpful feedback for our revisions. We gratefully acknowledge support from NSF via Grant Nos. NSF DUE-0136690 and DUE-0121679 under supervision of Edward Fox. Our thanks go to Kurt Maly, who heads the NCSTRL interface development team at Old Dominion University, for permission to publish a critical review of the system. We also are grateful to Xioming Liu, who works with Dr. Maly, and who provided us with cost estimates during our analysis.

We acknowledge the influence of ideas from Dr. Robert France, who has contributed for years to the areas of digital libraries and usability, among his other contributions to the world. We included some of his words about searching and browsing and believe he would have been a co-author of this paper, had his life not been tragically cut short last year. This paper is dedicated to the memory of Dr. Robert France.

References

- Andre T, Hartson HR, Belz S, McCreary F (2001) The user action framework: a reliable foundation for usability engineering support tools. *Int J Hum-Comput Stud* 54(1):107–136
- Bergmark D (2002) Models and tools for generating digital libraries: collection synthesis. In: Proceedings of the second ACM/IEEE-CS joint conference on digital libraries (JCDL '02). ACM Press, New York, pp 253–262
- Chen S-S, Choo C-Y (2002) A DL server with OAI capabilities: LOVE. In: Proceedings of the second ACM/IEEE-CS joint conference on digital libraries (JCDL '02). ACM Press, New York, p 388 (poster)
- Davis JR, Lagoze C (2000) NCSTRL: design and deployment of a globally distributed digital library. *J Am Soc Inf Sci* 51(3):273–280
- Dillon A (2002) Technologies of information: HCI and the digital library. In: Carroll J (ed) Human-computer interaction in the new millennium. ACM Press/Addison-Wesley, New York, pp 457–474
- Doward J, Reinke D, Recker M (2002) Preserving, securing, and assessing digital libraries: an evaluation model for a digital library services tool. In: Proceedings of the second ACM/IEEE-CS joint conference on digital libraries (JCDL '02). ACM Press, New York, pp 322–323
- France RA (2001) Personal communication
- Futrelle J, Chen S-S, Chang KC (2001) A CIS framework for NSDL. In: Proceedings of the first ACM/IEEE-CS joint conference on digital libraries (JCDL '01). ACM Press, New York, pp 124–125
- Geisler G, Giersch S, McArthur D, McClelland M (2002) Creating virtual collections in digital libraries: benefits and implementation issues. In: Proceedings of the second ACM/IEEE-CS joint conference on digital libraries (JCDL '02). ACM Press, New York, pp 210–218
- Glover EJ, Lawrence S, Gordon MD, Birmingham WP, Giles CL (2001) Web search – your way. *Commun ACM* 44(12):97–102
- Gupta A, Ludsher B, Moore RW (2002) Ontology services for curriculum development in NSDL. In: Proceedings of the second ACM/IEEE-CS joint conference on digital libraries (JCDL '02). ACM Press, New York, pp 219–220
- Hackman GS, Biers DW (1992) Team usability testing: are two heads better than one? In: Proceedings of the Human Factors Society 36th annual meeting. Human Factors and Ergonomics Society, Atlanta, GA, pp 1205–1209
- Hartson HR (2003) Cognitive, physical, sensory, and functional affordances in interaction design. Accepted for publication in *Behav Inf Technol* 22(5):315–338
- Hartson HR, Castillo JC (1998) Remote evaluation for post-deployment usability improvement. In: Proceedings of AVI'98 (advanced visual interfaces). ACM, L'Aquila, Italy, pp 22–29
- Hartson HR, Hix D (2003) Developing user interfaces – ensuring usability through product and process: short course in the usability engineering process. Unpublished Tutorial Notes
- Hartson HR, Shivakumar P (2002) Usability inspection report of NCSTRL (TR-02-08). Virginia Tech, Blacksburg, VA
- Hix D, Hartson HR (1993) Developing user interfaces: ensuring usability through product & process. Wiley, New York
- Janee G, Frew J (2002) Digital libraries for spatial data: the ADEPT digital library architecture. In: Proceedings of the second ACM/IEEE-CS joint conference on digital libraries (JCDL '02). ACM Press, New York, pp 342–350
- Klaus C, Andrew K (2001) An atmospheric visualization collection for the NSDL. In: Proceedings of the first ACM/IEEE-CS joint conference on digital libraries (JCDL '01). ACM Press, New York, p 463
- Lagoze C, Van de Sompel H (2001) The Open Archives Initiative: building a low-barrier interoperability framework. In: Proceedings of the first ACM/IEEE-CS joint conference on digital libraries (JCDL '01). ACM Press, New York, pp 54–62
- Lagoze C, Hoehn W, Millman D, Arms W, Gan S, Hillmann D, Ingram C, Kraft D, Marisa R, Phipps J, Saylor J, Terizzi C, Allan J, Guzman-Lara S, Kalt T (2002) Core services in the architecture of the national digital library for science education (NSDL). In: Proceedings of the second ACM/IEEE-CS joint conference on Digital libraries, Portland, Oregon, USA, pp 201–209
- Lagoze C, Terizzi C, Hoehn W, Millman D, Allan J, Guzman-Lara S, Kalt T, Arms W, Gan S, Hillman D, Ingram C, Kraft D, Marisa R, Phipps J, Saylor J (2002) Core services in the architecture of the National Science Digital Library (NSDL). In: Proceedings of the second ACM/IEEE-CS joint conference on digital libraries (JCDL '02). ACM Press, New York, pp 201–209
- Laleuf JR, Spalter AM (2001) A component repository for learning objects: a progress report. In: Proceedings of the first ACM/IEEE-CS joint conference on digital libraries (JCDL '01). ACM Press, New York, pp 33–40
- Landauer TK (1995) The trouble with computers: usefulness, usability, and productivity. MIT Press, Cambridge, MA
- Laurel B, Oren T, Don A (1990) Issues in multimedia interface design: media integration and interface agents. In: Proceedings of the CHI conference on human factors in computing systems. ACM Press, New York, pp 133–139
- Mayhew DJ (1999) The usability engineering lifecycle. Morgan Kaufmann, San Francisco
- Nanard M, Nanard J (2001) Cumulating and sharing end users' knowledge to improve video indexing in a video digital library. In: Proceedings of the first ACM/IEEE-CS joint conference on digital libraries (JCDL '01). ACM Press, New York, pp 282–289
- Nielsen J (1993) Usability engineering. Academic, Boston, MA
- Nielsen J, Mack RL (eds) (1994) Usability inspection methods. Wiley, New York
- Pomerantz J, Lankes RD (2002) Integrating expertise into the NSDL: putting a human face on the digital library. In: Proceedings of the second ACM/IEEE-CS joint conference on digital libraries (JCDL '02). ACM Press, New York, p 405

31. Powell AL, French JC (2000) Growth and server availability of the NCSTRL digital library. In: Proceedings of the fifth ACM conference on digital libraries. ACM Press, New York, pp 264–265
32. Theng YL (1999) Lostness and digital libraries. In: Proceedings of the fourth ACM conference on digital libraries. ACM Press, New York, pp 250–251
33. Van House NA (1995) User needs assessment and evaluation for the UC Berkeley electronic environmental library project. In: Proceedings of Digital Libraries '95: the second international conference on the theory and practice of digital libraries, pp 71–76. URL: <http://www.csdl.tamu.edu/DL95/papers/vanhause/vanhause.html>
34. Van House NA, Butler MH, Ogle V, Schiff L (1996) User-centered iterative design for digital libraries. *D-lib Magazine*. Retrieved 17 January 2003 from the World Wide Web: <http://www.dlib.org/dlib/february96/02vanhause.html>
35. Yaron D, Milton DJ, Freeland R (2001) Linked active content: a service for digital libraries for education. In: Proceedings of the first ACM/IEEE-CS joint conference on digital libraries (JCDL '01). ACM Press, New York, pp 25–32



How motivational orientation influences the evaluation and choice of hedonic and pragmatic interactive products: The role of regulatory focus

Marc Hassenzahl^{a,*}, Markus Schöbel^b, Tibor Trautmann^b

^a Economic Psychology and Human-Computer Interaction, University of Koblenz-Landau, Fortstraße 7, 76829 Landau, Germany

^b Department of Psychology and Ergonomics, Berlin University of Technology, F7 Marchstrasse 12, 10587 Berlin, Germany

ARTICLE INFO

Article history:

Received 7 March 2008

Received in revised form 18 May 2008

Accepted 27 May 2008

Available online 7 June 2008

Keywords:

User experience

Motivation

Affect

Hedonic quality

Product evaluation

ABSTRACT

The perceived quality of interactive products can be roughly divided into instrumental, task-related, pragmatic attributes (e.g., usefulness, usability) and non-instrumental, self-referential, hedonic attributes (e.g., novelty, beauty). Recent studies suggest that the weighting of both aspects in forming an overall evaluation of an interactive product heavily depends on features of the actual situation, such as whether an individual has to perform a specific task or not. The present paper extends these findings by assuming that a match between an individual's motivational orientation and particular product attributes (i.e., pragmatic, hedonic) moderates the perceived value of interactive products. Specifically, it shows how differences in regulatory foci (promotion or prevention focus), that is, differences in the way goal-directed behavior is regulated, influence product evaluation and choice. Participants were either set in a prevention focus (concern for safety and the avoidance of negative outcomes) or promotion focus (concern for personal growth and the attainment of positive outcomes). Subsequently, they were asked to evaluate and choose between a primarily pragmatic and a primarily hedonic mp3-player. The results revealed the expected effect of the activated regulatory focus on evaluation and choice. Individuals in a promotion focus rated the hedonic player as more appealing and chose it more frequently compared to individuals in a prevention focus. Reverse results, albeit not as strong, were found for the evaluation and choice of the pragmatic player. Our findings support the idea that product appeal and choice is strongly context-dependent. It further extends previous findings by showing that not only major differences in the situation, such as providing a specific task or not, impact product appreciation but that more subtle, motivational orientations can have similar effects.

© 2008 Elsevier B.V. All rights reserved.

1. Introduction

People perceive and evaluate interactive products along two different dimensions: *pragmatic* and *hedonic* quality (e.g., Hassenzahl, 2003; Batra and Ahtola, 1990). Pragmatic quality refers to the product's perceived ability to support the achievement of "do-goals," such as "making a telephone call," "finding a book in an online-bookstore" or "setting-up a webpage". In contrast, hedonic quality refers to the product's perceived ability to support the achievement of "be-goals," such as "being competent," "being related to others," "being special" (see Carver and Scheier, 1989 for more on "do-goals" and "be-goals"). Pragmatic quality calls for a focus on the product – its utility and usability in relation to a given task. It is about *how* one achieves a specific goal. Hedonic quality, however, calls for a focus on the Self, i.e., the question of *why* someone owns and uses a product. Here, more general human needs beyond the instrumental come into play, such as a need

for novelty and change, personal growth, self-expression and/or relatedness (see Rokeach, 1973; Ryan and Deci, 2000; Schwartz and Bilsky, 1987; Sheldon et al., 2001 for general lists of human needs and values).

In the context of interactive products, Hassenzahl (2003) distinguishes three different key drivers of hedonic quality: *Stimulation* (i.e., novelty and change, personal growth), *identification* (i.e., communication of identity to relevant others, relatedness) and *evocation* (i.e., keeping of memories, symbolizing) (see Malone, 1984; Logan et al., 1994; Jordan, 2000; Gaver and Martin, 2000 for alternative concepts). It is further assumed that people have implicit notions of the relation between particular product attributes (e.g., simple-complex, ordinary-novel) and pragmatic or hedonic quality, respectively (see Huffman and Houston, 1993; Reynolds and Olson, 2001). Simplicity, for example, may signal high pragmatic quality, whereas novelty may suggest high hedonic quality. Or to put it differently: Simplicity suggests the fulfillment of do-goals, whereas novelty suggests the fulfillment of be-goals.

Recent findings show that pragmatic and hedonic quality are perceived as unrelated. Furthermore, *both* contribute to the per-

* Corresponding author. Tel.: +49 179 1467726.

E-mail address: hassenzahl@uni-landau.de (M. Hassenzahl).

ceived overall value of a product (Hassenzahl et al., 2000; Hassenzahl, 2001). However, these studies tend to average across products and situations and, thus, do not account for contextual influences on product evaluation. In contrast, we assume that the relative weights of pragmatic and hedonic quality in forming an overall evaluation of a product are dependent on the context of evaluation (see Hartmann et al., 2007; Hassenzahl, 2003).

For instance, Hartmann and colleagues (2007) demonstrated the general context-dependency of preferences for Web sites. Students were told to have a look at three different HCI and design-related Web sites for either deciding for a summer internship or their Ph.D. research. A number of participants changed their preferences depending on the scenario. The majority indicated that the look and feel (i.e., hedonic) of the Web site was the most important attribute in the summer internship scenario, whereas the content of the Web site (i.e., pragmatic) was much more important for the Ph.D. research. A study by Hassenzahl et al. (2002) demonstrated that whether people have concrete do-goals to fulfill with a Web site or not influences the way the Web site is evaluated. Participants were either given tasks to accomplish (e.g., to find a particular information) or were instructed to "just have fun with the Web sites." For the task-group, pragmatic quality (e.g., simple–complex) was strongly correlated with the overall evaluation of the Web site (e.g., good–bad) (partial correlation = .87), whereas for the fun-group this correlation was not apparent (partial correlation = −.10). In the former case, usability as a product attribute was predictive for the product's overall evaluation, in the latter not. Depending on the context, the weight of usability in judging the overall appeal of the Web site has changed. With a similar manipulation, Rozendaal et al. (2007) found that increasing richness (i.e., complexity, variety, possibilities, that is, hedonic quality) in appearance and interaction with an interactive game resulted in increased engagement (i.e., enjoyment) for a fun-group but not for a task-group. For the latter group especially rich prototypes even created disengagement due to failures to accomplish the task. Moreover, Hassenzahl and Ullrich (2007) found that people in a fun-group base their evaluation of a product predominantly on the amount of spontaneity experienced while interacting, whereas the same aspect was negatively related to the product evaluation in a task-group. For the task-group, mental effort was strongly related to the evaluation of the product (i.e., less experienced mental effort led to a better evaluation of the product), an aspect that was irrelevant for the fun-group. With respect to this, the authors put forward the notion of *mode compatibility*, which is understood as a "compatibility of the way one ought to and actually does approach the product" (p. 436). People with an externally given task focus on the "how", i.e., the do-goal level and, thus, on the pragmatic quality of a product, whereas people without a task may rather focus on the "why", i.e., the be-goal level and, thus, on hedonic aspects.

So far, a number of recent studies demonstrated the context-dependency of product evaluation, i.e., situation-induced differences in the relative weights of particular product attributes. The present paper takes this a step further by showing that similar effects can be obtained by directly manipulating psychological states rather than the actual situation. To this end, we refer to the notion of two different *regulatory foci* in goal-directed behavior (Higgins, 1998, 2002). In a *promotion* focus (a particular motivational orientation), individuals are concerned with the presence or absence of positive outcomes. Behavior is motivated by ideals; individuals strive for personal growth and advancement. In a *prevention* focus, individuals attend to the absence or presence of negative outcomes. Behavior is motivated by oughts rather than ideals; individuals look for protection and security. We assume that the perceived value of a product is the consequence of a fit between the product's attributes and the actual regulatory focus. This

should be reflected by the situation-dependent weights given to attributes when evaluating or choosing a product.

A study by Safer (1998; cited in Higgins, 2002) provided first evidence for this notion. Individual foci strength was measured before participants were presented with two alternative cars. One car was described as predominantly luxurious (e.g., with plush soft leather seats) and neutral on a reliability dimension; the other was described as predominantly reliable (e.g., antilock brakes) and neutral on the luxury dimension. Safer found that prevention-focused participants, concerned with protection/oughts and seeking to avoid negative outcomes, preferred the reliable over the luxurious car. In contrast, promotion-focused individuals, concerned with advancement/ideals and seeking to gain positive outcomes, preferred the luxurious over the reliable car. Whereas Safer (1998) measured focus strength as a chronic variable, Florack et al. (2005) report a study, where regulatory focus was experimentally induced before participants had to choose between two products: a sun lotion, described as "protecting from sunburn," and a sun lotion stressing a "healthy tan". As predicted, people set in a prevention focus prior to choosing preferred the "protection" sun lotion, whereas people set in a promotion focus preferred the "healthy tan" sun lotion.

The present study attempts to link the notion of pragmatic and hedonic quality as independent aspects of interactive products to regulatory focus. Specifically, we suggest that prevention-focused individuals want to avoid failure and seek products, which offer effective ways to task accomplishment. As a consequence, they put more weight onto the pragmatic quality of a product and, thus, prefer primarily pragmatic products. In contrast, promotion-focused individuals rather concentrate on potential gains, stimulation and personal advancement. As a consequence, they put more weight on the hedonic quality of a product and, thus, prefer primarily hedonic products. To test these assumptions, we induced either a prevention or promotion focus and let participants choose between two different mp3-players. The players were either primarily hedonic or pragmatic. We expected an impact of regulatory focus on choice. In addition, we measured affective quality (Russell, 2003), strength of pragmatic and hedonic quality perceptions (Hassenzahl, 2001), as well as the general value (i.e., appeal) of each player. We expected that participant's perceived value of the mp3-players stems from the fit between regulatory focus and product attributes. In other words, whereas the perception of the product as primarily pragmatic or hedonic remains stable, the evaluation and the affective reactions to the product will be moderated by the regulatory focus.

2. Method

2.1. Participants

The study was conducted as an online experiment. Two-hundred fourteen individuals were invited via an email-list to participate in the study. Of those, 81 were either not responding or terminated the experimental task early. This left 133 responses (response rate: 62%) for further analysis. Participants (42% female, 58% male) had a mean age of 28 years (Min = 15, Max = 67) and used the computer for 32 h the week on average. All participants knew about software-based mp3-players; the majority (44%) indicated frequent use.

2.2. Procedure

After opening the online questionnaire, participants were informed that they participate in a study concerned broadly with the evaluation of software, consisting of a number of unrelated

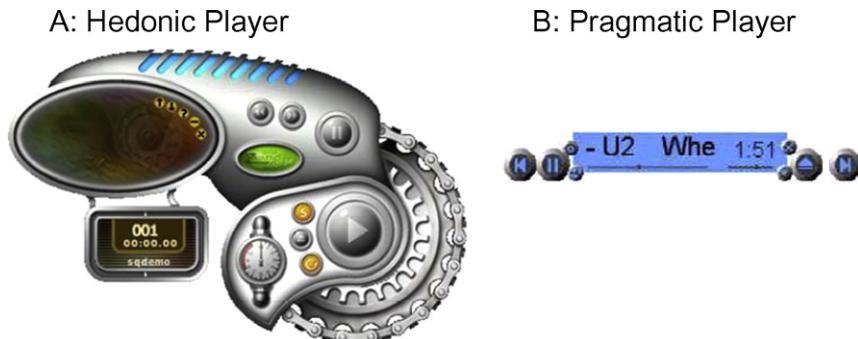


Fig. 1. Mp3-player skins pre-tested as primarily hedonic (A) and primarily pragmatic (B).

tasks. The experiment began with the measurement of the participants' current affective state. Participants then completed a "priming task," designed to induce a particular regulatory focus (promotion, prevention). After that, the current affective state was assessed again as a way to check the successful induction of the focus (i.e., manipulation check). Subsequently, participants were asked in a seemingly unrelated second task to evaluate and choose between two different mp3-players, one primarily hedonic, and the other primarily pragmatic. The affective quality of both players was measured, as well as pragmatic and hedonic quality perceptions, and an overall evaluation. Finally, demographic variables, computer usage and experience with mp3-players were assessed.

2.3. Independent variable: focus induction through a priming task

According to Friedman and Förster (2001) the activation of rudimentary semantics and procedural representations associated with striving for nurturance (promotion focus) or security (prevention focus) are sufficient to induce a particular regulatory focus. Based on this assumption, we confronted our participants with a decision problem in the "priming task". They were asked to imagine that they want to prepare a music compact disc as a birthday present for a good friend. They had only one spare disc left and were already late. Their choice was between a slow (4×) or a fast burning speed option (48×). By choosing the slower burning speed, they took the risk of arriving embarrassingly late for the birthday dinner. By choosing the higher speed, they ran the risk of a disc damage, that is, they will be on time but without birthday present. Upon their choice, they were shown a simulated progress bar and were then presented with particular outcomes depending on their experimental condition and their actual choice (4×, 48×). Participants in the *prevention* group were either told that the burning process failed and their last disc was lost (48×) or that the process took too long and that they will be embarrassingly late (4×). In both cases, the negative outcome was presented in a way, which emphasized social duties and responsibilities (i.e., bringing a present, being in time). Both will activate the concept of seeking safety and, thus, a prevention focus. Participants in the *promotion* group received the message that the burning process was successful (48×) or that they made it to the party on time (4×). The positive outcome was presented in a way, which emphasized individual performance, personal growth and approval by others (i.e., having the best present of all, being just in time). This will activate the concept of striving for nurturance and, thus, a promotion focus.

2.4. Dependent variables

2.4.1. A primarily hedonic and a primarily pragmatic product

In the present study, we let participants evaluate and choose between a primarily pragmatic and a primarily hedonic product.

We used mp3-player software (Sonique 1.63, see [http://en.wikipedia.org/wiki/Sonique_\(media_player\)](http://en.wikipedia.org/wiki/Sonique_(media_player))) with different "skins" as products. A "skin" is a graphic file used to change the appearance of an application's user interface. Sonique skins vary substantially in presentational style and usability (due to, for example, differences in layout, legibility, positioning of controls), while purpose and functionality remain constant. Overall, we pre-tested 12 different skins according to their pragmatic and hedonic quality. Thirty-one participants rated all skins on four seven-point semantic differential items with the verbal anchors *boring–interesting*, *ordinary–original* (i.e., hedonic quality) and *uncontrollable–controllable*, *complex–simple* (i.e., pragmatic quality) (see Section 2.4.4).

Each pair of items showed high internal consistency (hedonic: Cronbach's $\alpha = .81$; pragmatic: Cronbach's $\alpha = .81$). Based on this, we computed a mean hedonic and a mean pragmatic value for each skin. For our study we selected two skins (see Fig. 1), which differed maximally and significantly on their hedonic (skin A: Mean = 4.82, skin B: Mean = 2.83, $p < .001$) and pragmatic value (skin A: Mean = 3.23, skin B: Mean = 4.79, $p < .001$). In the remainder of the paper we refer to skin A as the (predominantly) hedonic player and B as the (predominantly) pragmatic player.

2.4.2. Affective state before and after focus induction

To verify the successful induction of the regulatory focus, the current *affective state* was measured before and after the actual focus induction. Affective state can be defined "as a neurophysiological state that is consciously accessible as a simple non-reflective feeling. [...] This feeling] is an integral blend of [valence] (pleasure–displeasure) and arousal (sleepy–activated) values" (Russell, 2003, p. 147). To measure affective state we used the nine-point, non-verbal *valence* and *arousal* items from the Self-Assessment-Manikin instrument (SAM, Bradley and Lang, 1994).

2.4.3. Affective quality

Affective quality (Russell, 2003) differs conceptually from current affective state (i.e., core affect). In contrast to affective state, affective quality is closely tied to an object and conceptualized as an object's perceived ability to impact and change affective states. Although both constructs differ in concept, Russell (2003) claimed that both can be assessed via the same two underlying dimensions – valence and arousal. Therefore, we used the SAM (Bradley and Lang, 1994) again to measure affective quality of the hedonic and pragmatic player after choice.

2.4.4. Measurement of the hedonic quality, pragmatic quality and appeal

Table 1 shows the items used to measure pragmatic quality, hedonic quality and appeal (i.e., general evaluation). Items are taken from Hassenzahl (2001).

A requirement for the valid and reliable assessment of pragmatic and hedonic quality is a high internal consistency of each

Table 1

Semantic differential items used to capture pragmatic quality (PQ), hedonic quality (HQ), and general evaluation (i.e., appeal) taken from Hassenzahl (2001)

Scale item	Anchors	
PQ 1	Comprehensible	Incomprehensible
PQ 2	Supporting	Obstructing
PQ 3	Simple	Complex
PQ 4	Predictable	Unpredictable
PQ 5	Clear	Confusing
PQ 6	Trustworthy	Shady
PQ 7	Controllable	Uncontrollable
PQ 8	Familiar	Strange
HQ 1	Interesting	Boring
HQ 2	Costly	Cheap
HQ 3	Exciting	Dull
HQ 4	Exclusive	Standard
HQ 5	Impressive	Nondescript
HQ 6	Original	Ordinary
HQ 7	Innovative	Conservative
APPEAL 1	Pleasant	Unpleasant
APPEAL 2	Good	Bad
APPEAL 3	Aesthetic	Unaesthetic
APPEAL 4	Inviting	Rejecting
APPEAL 5	Attractive	Unattractive
APPEAL 6	Sympathetic	Unsympathetic
APPEAL 7	Motivating	Discouraging
APPEAL 8	Desirable	Undesirable

Note: Order of items was randomized, some items were reversed, verbal anchors were in German.

scale combined with low scale intercorrelation. The latter is important to ensure that indeed different concepts were measured (i.e., discriminant validity). Internal consistency for the seven hedonic items was satisfactory (Pooled: Cronbach's $\alpha = .91$; hedonic player only: .78; pragmatic player only: .83). The same was apparent for the seven pragmatic items (Pooled: Cronbach's $\alpha = .95$; hedonic player only: .88; pragmatic player only: .82). Based on this, we calculated a hedonic and pragmatic scale value by averaging the hedonic and pragmatic items, respectively. As required, pragmatic and hedonic quality were not correlated for the pragmatic player ($r = .00$, n.s., $N = 133$) or for the hedonic player ($r = .09$, n.s., $N = 133$). In sum, the measurement of both qualities complies with requirements for reliable and factorial valid measurement.

General evaluation was measured with seven appeal items. Internal consistency of the scale was satisfactory (Pooled: Cronbach's $\alpha = .92$; hedonic player only: .94; pragmatic player only: .89). A mean appeal score was calculated as the average of the seven appeal items.

2.5. Predictions

With respect to product perception (i.e., hedonic, pragmatic) and evaluation (i.e., appeal), we first of all expect the hedonic player to be perceived as primarily hedonic and the pragmatic player to be perceived as primarily pragmatic by our participants. This is a necessary precondition for all remaining analyses.

We further assume that the perception of the products as either primarily hedonic or pragmatic is not affected by the induced regulatory foci (promotion, prevention). General evaluation (i.e., appeal), however, should be affected by foci, with higher appeal ratings given a fit between the product's attributes and the activated regulatory focus (i.e., promotion – primarily hedonic, prevention – primarily pragmatic).

The same holds true for affective quality–valence, which should as well be moderated by regulatory focus: a fit between focus and product attributes (e.g., promotion – primarily hedonic) will lead to more positive affective perceptions compared to a lack of fit (e.g., promotion – primarily pragmatic). This prediction parallels

the prediction for the general evaluation, which is due to the obvious overlap of affective quality and appeal.

In contrast, we assume affective quality–arousal not to be moderated by the activated regulatory focus. However, arousal and hedonic quality should be linked, because pleasurable stimulation is conceptualized as an important component of hedonic quality, captured by product attributes, such as *exciting*, *original* or *innovative* (Hassenzahl, 2003). This is supported by Desmet et al. (2001), who elicited the preferred affective quality of mobile phones. One group of participants preferred low arousal another group high arousal. A further interview (i.e., laddering) to explore the underlying reasons showed hedonic motives (freedom, high self-esteem) for the high arousal group and pragmatic motives (comfortable life, security) for the low arousal group. In sum, hedonic quality should be linked to psychological arousal.

As long as choice will depend on the perceived value of a product, we predict that changes in the valence of affective quality towards a product and its general appeal should also lead to changes in choice behavior. Specifically, we assume that the hedonic player should be more likely to be chosen by participants in a promotion focus compared to participants in a prevention focus.

3. Results and discussion

3.1. Manipulation check: focus induction

The induction of the prevention focus (by the “priming task”) was accompanied by a significant change in the valence of the affective state, $t(66) = 9.29$, $p < .001$. It dropped from the positive (Mean = 6.05, SE = 0.19) to a negative assessment of momentary affect (Mean = 3.64, SE = 0.17). In the promotion focus condition, momentary affect remained positive, $t(65) = 0.09$, n.s. (before: Mean = 5.94, SE = 0.20; after: Mean = 5.91, SE = 0.28). Both led to a significant difference in affect after the focus induction procedure, $t(131) = 7.00$, $p < .001$. One may argue that the induction procedure rather manipulated affective state (i.e., mood) than regulatory focus. However, none of the important dependent variables (affective quality–valence, appeal, and choice) were correlated with the valence of the affective state after the induction (correlations were in the range from .08 to -.01.), which makes this alternative explanation very unlikely.

As expected, arousal was not impacted significantly by the focus induction and no post-induction differences existed (after prevention: Mean = 5.78, SE = 0.24; after promotion: Mean = 5.68, SE = 0.25; $t(131) = -0.27$, n.s.). As above, neither affective quality–valence, appeal, nor choice were correlated with arousal (−.06 to .03). In sum, the manipulation was successful.

3.2. Manipulation check: products

A necessary precondition for further analyses is to make sure that both players were perceived as pre-tested, that is the hedonic player should be perceived as primarily hedonic and the pragmatic player as primarily pragmatic. To test this, we performed a $2 \times 2 \times 2$ analysis of variance with *player* (pragmatic, hedonic) and *quality aspect* (pragmatic, hedonic) as within-subject factors, *focus* (prevention, promotion) as a between-subject factor and the actual *value* of the hedonic and pragmatic scale as the dependent variable. We expected a disordinal interaction of *player* and *quality aspect*, without further interactions with regulatory focus. (Recall that quality perceptions should be independent from focus.)

Our analysis revealed a significant main effect of *quality aspect*, $F(1, 131) = 6.83$, $p < .01$, $\eta^2 = .05$, which was further qualified by a highly significant interaction of *quality aspect* with *player*,

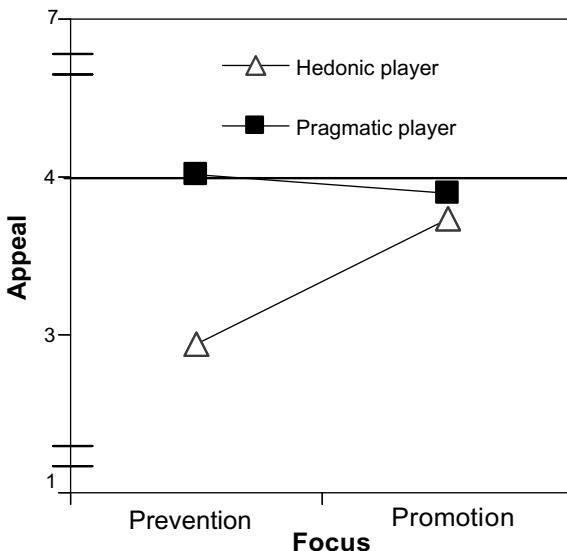


Fig. 2. Appeal (i.e., general evaluation) of the primarily hedonic and pragmatic player separately for the prevention and promotion focus.

$F(1,131) = 826.22, p < .001, \eta^2 = .86$. This interaction was disordinal with the hedonic player being perceived as primarily hedonic (hedonic: Mean = 5.15, SE = 0.07; pragmatic: Mean = 2.61, SE = 0.09) and the pragmatic player being perceived as primarily pragmatic (hedonic: Mean = 2.79, SE = 0.09; pragmatic: Mean = 5.49, SE = 0.08). None of the remaining main effects and interactions was significant. This shows that the players intended to be perceived as primarily hedonic or pragmatic are in line with the ratings of the participants.

3.3. Product evaluation

Fig. 2 shows the mean appeal ratings for the hedonic and pragmatic player for individuals in a prevention or promotion focus.

A 2×2 analysis of variance with *player* (pragmatic, hedonic) as a within-subject factor, *focus* (prevention, promotion) as a between-subject factor and *appeal* as the dependent variable re-

vealed significant main effects for *player*, $F(1,131) = 11.67, p < .001, \eta^2 = .08$, and *focus*, $F(1,131) = 9.27, p < .01, \eta^2 = .07$, which were further qualified by a highly significant *player* \times *focus* interaction, $F(1,131) = 6.74, p < .01, \eta^2 = .05$. As expected, promotion-focused participants rated the hedonic player as significantly more appealing compared to prevention-focused participants, diff = .81, $t(131) = 3.47, p < .01$. The reverse effect was apparent, but not significant for the pragmatic player, diff = -.13, $t(131) = -0.68, \text{n.s.}$

3.4. Affective quality

In general, arousal and valence ratings were neither correlated for the pragmatic nor for the hedonic player (pragmatic: $r = .12$, n.s., $N = 133$; hedonic: $r = -.14$, n.s., $N = 133$). Fig. 3 shows the mean ratings for arousal and valence for each player (pragmatic, hedonic) and each focus.

As expected, a 2×2 analysis of variance with *focus* (promotion, prevention) as a between-subject factor, *player* (pragmatic, hedonic) as a within-subject factor and *arousal* as the dependent variable revealed a highly significant main effect for *player* only, $F(1,131) = 161.71 p < .001, \eta^2 = 0.55$, with more arousal for the hedonic (Mean = 5.66, SE = 0.19) compared to the pragmatic player (Mean = 2.77, SE = 0.14). Confidence intervals (95%) showed arousal to be significantly above the center of the scale (5) for the hedonic (lower bound: 5.28) and below for the pragmatic player (upper bound: 3.06).

A further 2×2 analysis of variance with *focus* (promotion, prevention) as a between-subject factor, *player* (pragmatic, hedonic) as a within-subject factor and *valence* as the dependent variable revealed a highly significant main effect for *player*, $F(1,131) = 11.31, p < .001, \eta^2 = .08$, with a more positive reaction to the pragmatic (Mean = 5.45, SE = 0.13) compared to the hedonic player (Mean = 4.58, SE = .20). The main effect was further qualified by a highly significant *focus* \times *player* interaction, $F(1,131) = 7.38, p < .001, \eta^2 = .05$. The hedonic player was perceived as more positive by promotion-focused participants (Mean = 5.12, SE = 0.28) compared to prevention-focused participants (Mean = 4.05, SE = .28, $t(131) = 2.72, p < .01$), whereas the pragmatic player was perceived as – albeit not significantly – more positive by promotion-focused (Mean = 5.61, SE = 0.19) compared to prevention-focused participants (Mean = 5.29, SE = 0.19, $t(131) = -1.23, \text{n.s.}$).

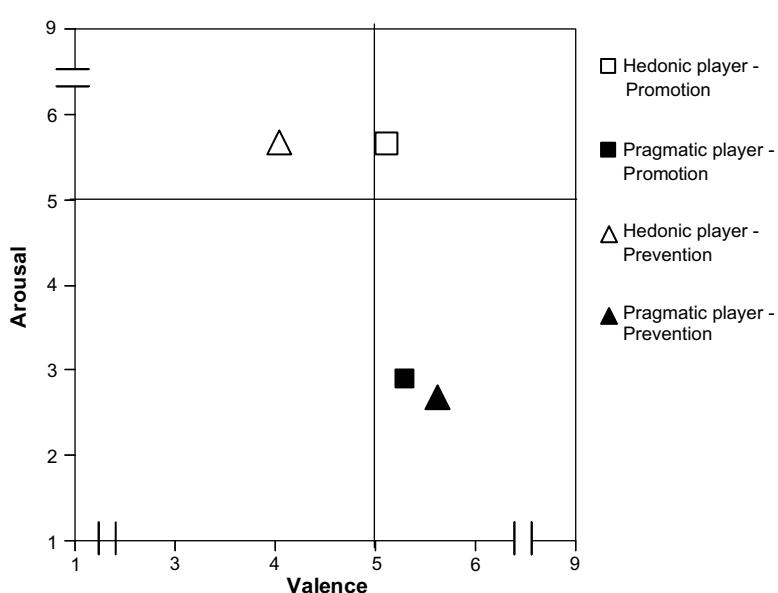


Fig. 3. Affective quality (valence, arousal) of the primarily hedonic and pragmatic player separately for the prevention and promotion focus.

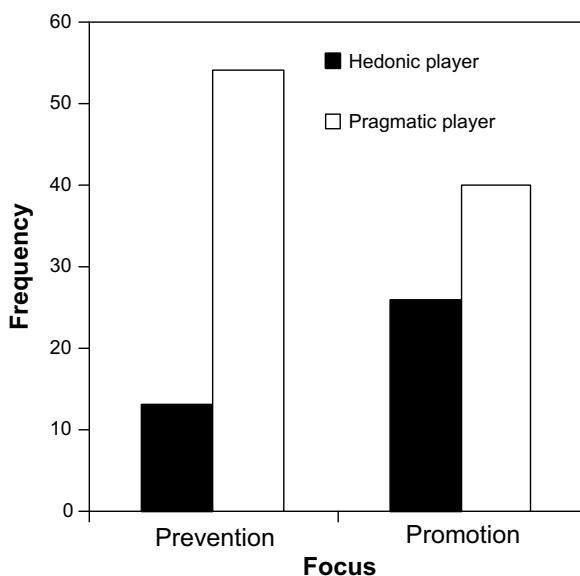


Fig. 4. Absolute frequency of choosing the primarily hedonic or pragmatic player in the prevention and promotion focus.

According to Russel's (2003) description of the interplay of valence and arousal, the induced focus changed the perceived affective quality of the hedonic player from 'exciting' to 'upsetting', whereas the pragmatic player remained 'calm'.

3.5. Choice

Fig. 4 shows the absolute frequency of choices of the pragmatic or hedonic player in either prevention or promotion focus.

In general, the pragmatic player was preferred over the hedonic (94 of 133, 71%, $\chi^2(1) = 22.74, p < .001$). In addition, there was the expected effect of focus on choice, with more hedonic (and less pragmatic) choices in the promotion compared to the prevention focus, $\chi^2(1) = 6.41; p = 0.01$. Promotion-focused participants chose the hedonic player twice as often as prevention-focused participants.

4. Summary and conclusion

The present study demonstrates how different motivational orientations, namely different *regulatory foci*, impact choice, evaluation and perceived affective quality. A closer look at the findings, however, reveals a certain asymmetry: Whereas the effect of regulatory focus on the perceived appeal and affective quality of the primarily hedonic player was significant, all observed differences of the evaluation of the primarily pragmatic player were in the predicted direction, but not as strong. Strictly speaking, the hedonic player was more prone to be devalued if participants were in an incompatible mode (i.e., prevention mode) compared to the pragmatic player. Whether this is a generalizable outcome stemming from inherent differences in the way hedonic or pragmatic products are perceived or whether it is just a matter of differences in the strength of the effect stemming from the products (e.g., mp3-players mainly fulfill pragmatic functions) must remain open. However, despite this slight limitation the present findings are nevertheless able to make a crucial point – to demonstrate that motivational orientations impact product evaluation and choice. This suggests that perceived value depends on the fit between product character (primarily hedonic, primarily pragmatic) and regulatory focus (promotion, prevention).

Generally, the pragmatic player was preferred over the hedonic player (e.g., main effect on appeal, main effect on choice). This may be due to a priori differences in the appeal of both players. An alternative explanation, however, refers to the particular judgmental situation in the present study. Hsee (1996) argued for a fundamental difference between situations in which two options are evaluated either separately or jointly. Okada (2005) applied this argument to the choice between hedonic and utilitarian consumer goods. Specifically, she showed that in case of a separate evaluation a hedonic product will be rated as more appealing than an utilitarian product of the same monetary value. However, in a joint evaluation situation the utilitarian product is preferred. In the present study, the same effect may have led to a better evaluation and more frequent choice of the pragmatic player. In other words, the observed general differences in the evaluation and choice of the players may be rather due to the particular situation (i.e., joint evaluation in a choice situation) than to a true, a priori difference of the appeal of both players.

Another interesting finding is the difference in affective quality-arousal between the hedonic and pragmatic player. It was argued that experiencing pleasurable stimulation is an important be-goal and that product attributes such as novelty or originality (i.e., hedonic quality) signal potential fulfillment of this goal. The present study establishes the link between specific product attributes (e.g., novelty) and expected psychological outcomes (i.e., arousal) as predicted by the notion of hedonic quality, and, thus lends credit to the underlying model.

In sum, the present results consistently support the idea that product appeal and choice is strongly context-dependent. In our study, the regulatory foci were situationally induced by a priming task, and it was shown that these foci impact a subsequent, seemingly unrelated evaluation and choice task. In real situations, the existence and impact of regulatory foci might be rather thought of as a natural response to particular situational cues. The motivational orientation adapts goal attainment and related behavior to the situation at hand. Therefore, motivational orientations may then be thought of as "shortcuts", which free us from the burden of considering a large variety of diverse, potentially occurring situations. If we better understand, how different motivational orientations color product evaluation, choice and use, we may be able to make some fundamental design decisions holding for a wide variety of situations, without actually having to know and to describe these situations in detail.

The most important message of the present paper for practitioners and researchers in the field of HCI and user-centered design is to abandon the still widespread idea that the perceived value of an interactive product is stable, given its fit into a particular context of use. We must embrace the idea that value is generated in the moment of the human-product interaction, and that different qualities may have changing appeal depending on a person's state or motivational orientation. Design may anticipate that, but more importantly product evaluation has to learn its twofold lesson, namely that (1) general measures of value, appeal, or satisfaction may be fleeting and less reliable than measures of product perception, such as a measure of pragmatic quality, and that (2) particular product attributes are not per se more important than others. Especially, the latter may be hard to accept as long as it implies that a quality such as usability is not a necessary precondition for appeal as, for instance, Jordan's (2000) hierarchical concept of utility, usability and pleasure suggests. Usability is important, if people focus on the attainment of do-goals or the presence and absence of negative outcomes. If in a given moment, they rather care about particular psychological needs, such as attaining pleasurable stimulation, other, hedonic qualities will become more important. Focusing on pragmatic qualities alone will inevitable fail to produce appealing interactive products in the same way a focus on he-

donic quality alone will fail. The challenge is to consider both qualities and to find a balance. Understanding the link between particular product attributes and context-dependent prioritizations of individual needs and motives (Hassenzahl, 2006; Sheldon et al., 2001) is an important step towards this goal, taken by the present study. Future studies will dive deeper into the impact of psychological states on product evaluation and choice.

References

- Batra, R., Ahtola, O.T., 1990. Measuring the hedonic and utilitarian sources of consumer choice. *Marketing Letters* 2, 159–170.
- Bradley, M.M., Lang, P.J., 1994. Measuring emotion: the self-assessment manikin and the semantic differential. *Journal of Behavior Therapy and Experimental Psychiatry* 25, 49–59.
- Carver, C.S., Scheier, M.F., 1989. On the Self-Regulation of Behavior. Cambridge University Press, New York.
- Desmet, P.M.A., Overbeeke, C.J., Tax, S.J.E.T., 2001. Designing products with added emotional value: development and application of an approach for research through design. *The Design Journal* 4, 32–47.
- Florack, A., Scarabis, M., Gosejohann, S., 2005. Regulatory focus and consumer information processing. In: Kardes, F.R., Herr, P.M., Nantel, J. (Eds.), *Applying Social Cognition to Consumer-Focused Strategy*. Lawrence Erlbaum Associates, Mahwah, NJ, pp. 235–263.
- Friedmann, R.S., Förster, J., 2001. The effects of promotion and prevention cues on creativity. *Journal of Personality and Social Psychology* 81, 1001–1013.
- Gaver, W.W., Martin, H., 2000. Alternatives. Exploring information appliances through conceptual design proposals. In: Turner, T., Szwilus, G. (Eds.), *Proceedings of the CHI 2000 Conference on Human Factors in Computing*. ACM, Addison-Wesley, New York, pp. 209–216.
- Hartmann, J., Sutcliffe, A., De Angeli, A., 2007. Investigating attractiveness in web user interfaces. In: *Proceedings of the CHI 2007 Conference on Human Factors in Computing*. ACM, Addison-Wesley, New York, pp. 387–396.
- Hassenzahl, M., 2001. The effect of perceived hedonic quality on product appealingness. *International Journal of Human-Computer Interaction* 13, 479–497.
- Hassenzahl, M., 2003. The thing and I: understanding the relationship between user and product. In: Blythe, M., Overbeeke, C., Monk, A.F., Wright, P.C. (Eds.), *Funology: From Usability to Enjoyment*. Kluwer, Dordrecht, pp. 31–42.
- Hassenzahl, M., 2006. Hedonic, emotional, and experiential perspectives on product quality. In: Ghaoui, C. (Ed.), *Encyclopedia of Human-Computer Interaction*. Idea Group, pp. 266–272.
- Hassenzahl, M., Kekez, R., Burmester, M., 2002. The importance of a software's pragmatic quality depends on usage modes. In: Luczak, H., Cakir, A.E., Cakir, G. (Eds.), *Proceedings of the 6th international conference on Work With Display Units (WWDU 2002)*. ERGONOMIC Institut für Arbeits- und Sozialforschung, Berlin, pp. 275–276.
- Hassenzahl, M., Platz, A., Burmester, M., Lehner, K., 2000. Hedonic and ergonomic quality aspects determine a software's appeal. In: Turner, T., Szwilus, G. (Eds.), *Proceedings of the CHI 2000 Conference on Human Factors in Computing*. ACM, Addison-Wesley, New York, pp. 201–208.
- Hassenzahl, M., Ullrich, D., 2007. To do or not to do: differences in user experience and retrospective judgments depending on the presence or absence of instrumental goals. *Interacting with Computers* 19, 429–437.
- Higgins, E.T., 1998. Promotion and prevention: regulatory focus as a motivational principle. In: Zanna, M.P. (Ed.), *Advances in Experimental Social Psychology*. Academic Press, San Diego, pp. 1–46.
- Higgins, E.T., 2002. How self-regulation creates different values: the case of promotion and prevention decision making. *Journal of Consumer Psychology* 12, 177–191.
- Hsee, C.K., 1996. The evaluability hypothesis: an explanation for preference reversals between joint and separate evaluations of alternatives. *Organizational Behavior and Human Decision Processes* 67, 247–257.
- Huffman, C., Houston, M.J., 1993. Goal-oriented experiences and the development of knowledge. *Journal of Consumer Research* 20, 190–207.
- Jordan, P., 2000. *Designing Pleasurable Products. An Introduction to the New Human Factors*. Taylor & Francis, London, New York.
- Logan, R.J., Augaitis, S., Renk, T., 1994. Design of simplified television remote controls: a case for behavioral and emotional usability. In: *Proceedings of the 38th Human Factors and Ergonomics Society Annual Meeting*. HFES, Santa Monica, pp. 365–369.
- Malone, T.W., 1984. Heuristics for designing enjoyable user interfaces: lessons from computer games. In: Thomas, J.C., Schneider, M.L. (Eds.), *Human Factors in Computer Systems*. Ablex, Norwood, NJ, pp. 1–12.
- Okada, E.M., 2005. Justification effects on consumer choice of hedonic and utilitarian goods. *Journal of Marketing Research* 42, 43–53.
- Reynolds, T.J., Olson, J.C., 2001. *Understanding Consumer Decision Making: A Means End Approach to Marketing and Advertising Strategy*. Lawrence Erlbaum, Mahwah, NJ.
- Rokeach, M., 1973. *The Nature of Human Values*. Free Press, New York.
- Rozendaal, M.C., Keyson, D.V., Ridder, H.D., 2007. Product behavior and appearance effects on experienced engagement during experiential and goal-directed tasks. In: *Proceedings of the International Conference on Designing Pleasurable Products and Interfaces (DPPi2007)*, pp. 181–194.
- Russell, J.A., 2003. Core affect and the psychological construction of emotion. *Psychological Review* 110, 145–172.
- Ryan, R.M., Deci, E.L., 2000. Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. *American Psychologist* 55, 68–78.
- Safer, D.A., 1998. Preferences for luxurious or reliable products: promotion and prevention focus as moderators. Columbia University, Department of Psychology.
- Schwartz, S.H., Bilsky, W., 1987. Toward a universal psychological structure of human values. *Journal of Personality and Social Psychology* 3, 550–562.
- Sheldon, K.M., Elliot, A.J., Kim, Y., Kasser, T., 2001. What is satisfying about satisfying events? Testing 10 candidate psychological needs. *Journal of Personality and Social Psychology* 80, 325–339.

UserX Story: Incorporating UX Aspects into User Stories Elaboration

Joelma Choma^(✉), Luciana A.M. Zaina, and Daniela Beraldo

Federal University of São Carlos, Sorocaba Campus, São Paul, Brazil

j.h.choma@hotmail.com, lzaina@ufscar.br,
danielaberaldo12@gmail.com

Abstract. In the last decade, many strategies have been employed successfully to incorporate User Experience (UX) practices into agile development in order to increase user satisfaction with the product. In this paper, we present a grammar for stories of interaction called UserX Story, in order to remedy the difficulties encountered by teams to insert UX aspects and usability requirements in the first steps of software conception. An action research approach was applied to carry out the research within the industry, allowing us to work closely with the agile teams. The research cycle was split in three steps. In the first step, we carried out a technical literature survey, aiming to investigate the use of user stories in the agile methodologies, and additionally, an ethnographic study also was carried out in order to understand how the traditional user stories were being developed by product owners. In the second step, we proposed together with both teams - UX and agile - a grammar to UserX Story incorporating two concepts of UX: personas and Nielsen's heuristics. In the third step, six product owners were invited to implement the UserX Stories in real projects. After that, we interviewed the participants aiming to collect their experiences with the implementation of UserX Stories. Thus, we have found out that most of the agile teams approved the use of the stories incorporating UX aspects.

Keywords: Agile · Scrum · User experience · Usability · User story

1 Introduction

Agile software development methodologies, such as Scrum, are being adopted by many software companies, due to their lightweight and adaptive nature, seeking mainly deal with the lack of predictability that is common within the traditional development process. According to agile principles the highest priority is to satisfy the customer through early and continuous delivery of valuable software [13].

Nonetheless, due to lack of awareness about usability issues during agile development, the focus is not specifically on creating a design with good usability [11]. Customers often focus in core functionalities. Especially when customers are involved in providing constant feedback during software development, they are able to verify, for instance, the ease of use of a system under development [25]. However, the usability can be better assessed from the end-user experience perspective [23].

Agile approaches emphasize different design styles, once the quality of design is essential to maintaining agility. Design is considered a continuous activity that should be performed throughout the software development process rather than to be an entirely up-front activity [13]. The interest of integrating the User Experience Design (UXD) into agile methodology practices has been increased in the last decade in order to provide high quality user experience, and usability as an important item to add more value to the software [18].

Many strategies have been employed, including some UX professionals who have found ways to incorporate successfully UXD into their agile projects [2, 30]. However, few proposals concern to incorporation of design methods and more suitable artifacts to support the communication between designers and agile teams working closely. Artifacts in an agile environment are important mechanisms to maximize the transparency of information, and support decisions during the software development [27], while the UXD artifacts are produced to outline useful solutions from understanding users' needs, tasks, and context of use [3].

Considering that the shared use of artifacts by both teams (designers and programmers) is an important instrument to support the integration of the UXD methods into agile practices, this paper presents a proposal to the incorporation of UX aspects including usability recommendations in the elaboration of user stories, from interaction scenarios. User story is one of the most popular artifacts for specifying and communicating agile requirements. In most cases, however, the user stories do not address usability aspects, and take no account of real users' needs and behaviors [15]. Recently, in the course of a research project in a software industry, we detected that the Scrum teams were having difficulties to insert UX aspects in their stories to define user interaction requirements. In this paper, we describe the solution found to help practitioners to write user interaction stories called UserX Stories.

The remainder of this paper is organized as follows: Sect. 2 introduces the background about agile requirements engineering, including usability aspects; Sect. 3 some considerations about methodology, and research issues; Sect. 4 presents the research context, the steps to build and evaluate the artifact adapted from user stories, discussing the outcomes found in practice; and finally Sect. 5 presents the conclusions and points out directions for further works.

2 Background

Agile development is more flexible concerning elicitation and management of requirements than traditional software development, for the purpose of a quick reaction to changes in order to match customer needs [13]. The main practices used in the agile methodologies to transfer ideas from the customer to the development team are the face-to-face communication and frequent feedback [26]. These practices allow the adaptation of requirements that are discussed in detail with the customers during the development process. The agile requirement specification is not centralized in one phase before development; instead, this activity is evenly spread throughout development [4]. Rather than written specification for creating extensive requirements documents, agile

methodologies usually adopt more simple techniques. In Scrum software projects [27], the most popular agile methodology, the user stories are artifacts adopted to define requirements in a high-level [8].

User stories are defined focusing on customer value, unlike other forms of requirements specification, which focus on system operations [25]. User stories are written throughout the agile project in a common language, intelligible to the users, in an effort to keep the attention and awareness on the needs of the users to emphasize their goals [8]. Everyone on the agile team participates in the writing of the stories with the goal of creating the list of features (product backlog) that describe the functionalities to be added over the course of the project [7]. Once that user stories contain just enough information to drive the development, more details about the requirements can be exploited by the development team through a conversation directly with the customer and/or other stakeholders [16]. Acceptance criteria are commonly added to user stories to guide the acceptance tests, in order to verify whether the stories were developed exactly how the customer expects [7].

According to Ramesh et al. [25] and Inayat et al. [16], a major concern about the iterative requirements engineering in agile development is the inadequate attention given to non-functional requirements - often ill-defined and ignored - during early development cycles. Overall, software engineers deal with usability as a non-functional requirement, believing that it can be considered later in the software development [28].

Usability requirements are qualitative attributes that specify user effectiveness, efficiency, or satisfaction levels that the system should achieve [23]. Some studies have highlighted the relationship between usability and functional requirements, aiming identify functional usability recommendations [19, 20]. According to these studies, usability features, especially those with functional implications, should be dealt as early as possible and included along with all the other requirements features, in order to provide a quality user experience and to avoid any rework and further unnecessary costs due to required adjustments.

3 Methodology and Research Issues

The last years, we have developed a project in partnership with a software industry to integrate UX in their software development process. This company has more than thirty years of experience in the development of ERP (Enterprise Resource Planning) systems for several market segments. Besides mechanisms and techniques to improve the usability of the ERP systems, other objective of this project has been to propose more suitable artifacts to support the integration of UX concepts within Scrum projects, agile method recently adopted by the company. Aiming to improve the communication of usability issues between UX designers and Scrum teams, we proposed previously a protocol for communication of design solutions and usability recommendations, whose findings can be seen in [5]. In this paper we present a new proposal of artifact incorporating UX aspects in user stories.

In order to carry out the research within the software industry, allowing us to work closely with the agile teams, we have applied a research approach called SoftCoDeR

(Software Cooperative Design Research) [6]. This research approach is based on a qualitative method of Action Research called Cooperative Method Development (CMD) proposed by Dittrich et al. [10], matched with the Design Science Research framework proposed by Hevner et al. [14]. On one hand, CMD approach provides a structured process of research, in which three phases are defined: (1) Understanding Practice (qualitative empirical investigations), (2) Deliberate Improvements (aiming to solve the problems identified in the first phase) and (3) Implement and Observe Improvements (checking the effectiveness of the improvements). On the other hand, DSR provides the guidelines to design artifacts of value based on both real need of industry (relevance) and scientific knowledge (rigor of research).

In the following section, we describe the research context and the three phases of the SoftCoDeR approach that was applied to answer the specific research question concerning to the building and evaluating of the artifact adapted from user stories, integrating UX aspects elicited from interaction scenario.

4 UX Aspects on User Stories Elaboration

The aforementioned protocol for communication of design solutions and usability recommendations had been proposed specifically to supply a need of the designers of a formal procedure to report results of user testing to Scrum teams. The proposed protocol was based on two UX concepts: personas and Nielsen's heuristics. Both UX concepts were primarily used to establish a common vocabulary among the developers, testers and UX designers.

Personas have been widely employed in the fields of usability and user experience as an artifact to describe groups of typical users by the creation of hypothetical archetypes. The personas can help create an empathy with users, facilitating the understanding on characteristics, behaviors and their deeper needs [24]. In addition to this, personas also can help in the development of interaction scenarios, and/or to describe tasks to the usability testing planning [9].

Nielsen's heuristics are one of the most used usability guidelines for user interface design, developed by Nielsen and Molich in the early 90's [22]. Also, these guidelines are commonly used to support the critical analysis in usability inspections (heuristic evaluation), and to check the problems identified in usability testing [12].

We have observed in practice that such concepts - personas and Nielsen's heuristics - can to improve the level of awareness and concerns on usability aspects of UX designers and developers (programmers and testers) [5]. The incorporation of both concepts to report the usability problems identified in user tests increased the level of reliability of the agile teams regarding the recommendations proposed by designers. However, we observed that the product owners (POs) were having difficulties in understanding such usability concepts, and also they did not know how to incorporate UX issues in the product requirements.

Considering that the POs were most familiarized with user stories to deal with agile requirements, we have suggested an adaptation of this artifact incorporating the vision of the user experience by employing the same UX concepts used on protocol for

communication of usability recommendations. Thereby, a research cycle was started in order to answer the following question:

RQ: How could personas and Nielsen's heuristics concepts be incorporated into the User Stories?

4.1 Understanding Practice

In the first phase, we have carried out (i) an ethnographic study in order to understand how the user stories were being developed by POs; and (ii) a technical literature survey, aiming to investigate the use of user stories in agile practices.

Regarding user stories development (i), we found out that these were being written in a more traditional way, as the template popularized by Mike Cohn [7], shown in Fig. 1. Such user stories were stored in a software used for issue tracking and project management, instead of written on index cards or sticky notes.

As a <type of user>, I want <some goal> so that <some reason>

Fig. 1. User Story template

As for the results of literature review (ii), we found some recommendations about how to deal with usability issues and user stories together. Moreno and Yagüe [20], for instance, have identified three ways to incorporate functional usability requirements with user stories: (1) adding new stories to represent the requirements that are directly derived from usability (called usability stories); (2) adding or modifying tasks in US (detailing as needed); and (3) adding or modifying acceptance criteria. According to these authors, there are usability recommendations that: have positive impact on the final quality of use of software systems; can be considered as functional usability requirements that complement the traditional requirements; and can be documented as user stories - the usability stories - because both are similar. On the other hand, Barksdale et al. [1] have used conceptual maps to design a complete picture to the user, detailing the connections between user stories and scenarios. Scenarios are a type of design artifact commonly used to describe how a particular user uses the system for a specific task, considering the context and environment where it will be held [17]. The conceptual mapping was proposed to mitigate existing conflict between designers and agile teams to communicate usability and interaction requirements during the agile project. Still on scenarios, Sohaib and Khan [29] also recommended the use of them along with user stories during the exploration phase, and in addition, heuristic evaluation during acceptance testing.

4.2 Deliberate Improvements

Bearing in mind the research question, and from the outcomes of the literature review and the ethnographic study, we proposed to the teams a grammar - incorporating personas and Nielsen's heuristics - to describe interaction stories, that we called UserX

Story. The UserX Stories should be written from scenarios associated with user's actions and respective feedback necessary to achieve goals supported by the system.

Figure 2 shows the first version proposed to the UserX Story, whose traditional grammar of the user story was modified, by replacing (1) <type of user> by <personas>, to provide a clear target for developers to focus on; and (2) <some reason> by <Nielsen's heuristic(s)>, to highlight, from usability point of view, the positive impact on user interaction to achieve their goals.

<i>UserX Story (template)</i>
As a < Persona >, I want/need < goal> so that < Nielsen's heuristic> will be met

Fig. 2. Initial proposal to UserX Story template (1st version)

The initial proposal was discussed between the researchers, POs, Scrum Master and UX designer. From the discussions between researches and practitioners, the artifact had been improved, resulting in a second version to the UserX Story. In the second version (Fig. 3), the user stories would be expanded from the outcomes of the data collected in the user research phase. User research focuses on understanding user behaviors, needs, and motivations through observation techniques, task analysis, and other feedback methodologies, e.g. the usability testing and the recommendations pinpointed in the aforementioned communication protocol.

<i>UserX Story (template)</i>
As a < Persona >, I want/need < goal>, for this <interaction>, through/ when [<task> / <context>]. I evaluate that my goal was achieved when <feedback>
<i>Acceptance criteria:</i>
Checks <action> through <set of conditions> to satisfy <Nielsen's heuristic(s)> of action, and < Nielsen's heuristic(s)> of feedback.

Fig. 3. UserX Story template (2nd version)

The UserX Stories are told from the perspective of the persona, who needs a particular condition for interaction. Such a condition can meet multiple personas. The stories describe an interactive process wherein the persona has a goal to achieve, for this s/he acts on the interface (interaction), to perform tasks (steps /features to effect the action) in a particular context (usage pattern). The persona will assess whether the objective was achieved interpreting system feedback. Aiming to verify whether stories were developed such that it exactly met the user interaction needs, the acceptance criteria

should describe the action, the set of conditions, and the Nielsen's heuristics (action/feedback) that will be satisfied once the goal is successfully achieved.

The workshop entitled "Interaction Stories" was organized in order to make a warm up for the creation of stories adding the vision of UX through the use of personas and Nielsen's heuristics. Table 1 shows some information about the six POs who attended the workshop. All the participants had more than ten years of experience in the IT field. However, five out of the six participants had little experience with agile methodology (less than a year).

Table 1. Overview of the workshop participants

Product Owner	Experience in the IT field	Experience in the Company	Experience with Agile Methodology
P1	14 years	13 years	6 months
P2	11 years	2 weeks	2 weeks
P3	15 years	8 years	6 months
P4	15 years	9 years	9 months
P5	27 years	27 years	3 months
P6	17 years	3 years	3 years

The workshop was divided into (i) explanation of the concept of user story, personas and Nielsen's heuristics (ii) presentation of the UserX Story; and (iii) an exercise of writing stories from the proposed template, including the acceptance criteria. Some data collected with end-users during a workshop about "User Research" techniques previously performed with another group of participants, were provided as supplementary material to support the proposed exercise. Some personas were also included in this material. At the end of the workshop activities we discussed with the POs the next step for implementing the interaction stories in real projects, in order to evaluate the proposed artifact.

4.3 Implement and Observe Improvements

In this phase, the POs had one month to implement the UserX Story into one of their projects. After this period, researchers carried out individuals' interviews with the POs to collect their experiences with the implementation of UserX Stories. Figure 4 shows an implemented UserX Story for the redesign of part related for the issuance of reports for a Tax Bookkeeping sub-module.

Most POs had shared the UserX Story with their respective Scrum teams, which reacted positively, approving the use of the interaction stories, as well as the proposed template. However, two POs had not implemented the UserX Stories in their projects, since they were working on small changes that were related exclusively to business rules (legal requirements), and such changes would have had no impact on user interaction.

As we have discussed with company's practitioners, further tests will be needed to evaluate the procedures in which user stories are written from items reported in the protocol for the communication of usability recommendations. However, it was suggested that, firstly, the items reported in the protocol should be discussed between

UserX Story - Tax Bookkeeping sub-module	
<p>As a <Leo Walker> I need to <issue financial reporting and balance sheets, filtered by agents>, for this < the system allows me to choose the agent that I want to filter >, through/ when [<for issuing the report> / < regardless of the organization to which I am placed in the system, it being subsidiary or consolidator>]. I evaluate that my goal was achieved when <the report only listed the launches carried for the selected agent ></p>	
<p>Acceptance criteria:</p> <p>Checks < the system will validate if that agent code can be used for the selected organization > through < filtering by agent code > to satisfy <H5> of action, and < H9> of feedback.</p> <p>Checks < the system should display the agent name next to the chosen code > through < choosing an agent, either by agent code or searching > to satisfy <H1> of action, and <H6> of feedback.</p>	

Fig. 4. UserX Story and acceptance criteria implemented by a PO

UX designer and Scrum team. And then, whether the Scrum team agrees with the recommended item, the item would be written in UserX Story template with the participation of the UX designer. Otherwise, the recommendation is discarded if, for example, there are technical limitations preventing it to be implemented.

One issue identified during interviews was related to the level of detail that acceptance criteria should be written. One of the POs commented that your criteria seemed quite detailed; thinking that reaching a greater level of detail should be the role of testers. Generally, acceptance criteria should be detailed enough to define when the user story is satisfied. Nonetheless, it is worth noting that there is some confusion about acceptance criteria and test cases. According Nazzaro & Suscheck [21], acceptance criteria should answer the question, “How will I know when I am done with the story?” and test cases answer the questions, “How do I test and what are the test steps?” Therefore, test cases can require more detail than acceptance criteria. Remembering that, the main focus of the acceptance criteria in UserX Story is satisfying usability guidelines.

5 Conclusion and Further Work

The outcomes of this work are part of a set of actions performed together with professionals from a company developing ERP systems, whose main objective is the incorporation of interaction design into the software development agile process.

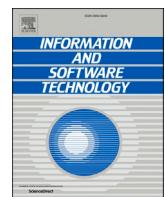
This paper presents a solution to incorporate UX aspects into user stories, aiming to guide and facilitate the work of the agile teams in terms of usability requirements. The UserX Stories are told from the perspective of the persona representing a group of users; and the heuristics are intended to reinforce the acceptance criteria to highlight the positive impact of user interaction if the conditions were satisfied.

Although the proposal of interaction stories has already been approved by the majority of company's agile teams, more research is required in this area through the involvement of further organizations and agile teams. We found out that further tests will be needed to evaluate the cycle in which UserX Stories are written from results reported in usability tests. In the further works, we intend to draw up a set of guidelines to target the use of the UserX Stories. Another issue deserving further research refers to the granularity of the acceptance criteria.

References

1. Barksdale, J., Ragan, E., McCrickard, D.: Easing Team Usability: A Concept Mapping Approach. Agile Conference. IEEE, Chicago (2009)
2. Brown, D.D.: Five Agile UX Myths. *J. Usability Studies* **8**(3), 55–60 (2013)
3. Brown, J.M., Lindgaard, G., Biddle, R.: Collaborative events and shared artefacts: Agile interaction designers and developers working toward common aims. In: Agile Conference AGILE 2011, pp. 87–96. IEEE Computer Society, Salt Lake City (2011)
4. Cao, L., Ramesh, B.: Agile requirements engineering practices: an empirical study. *IEEE Softw.* **25**(1), 60–67 (2008)
5. Choma, J., Zaina, L.A., Beraldo, D.: Communication of design decisions and usability issues: a protocol based on Personas and Nielsen's heuristics. In: Kurosu, M. (ed.) Human-Computer Interaction. LNCS, vol. 9169, pp. 163–174. Springer, Heidelberg (2015)
6. Choma, J., Zaina, L. A., Silva, T.S.: Towards an approach matching CMD and DSR to improve the Academia-Industry software development partnership. In: Brazilian Symposium on Software Engineering (SBES). IEEE (2015)
7. Cohn, M.: User Stories Applied: For Agile Software Development, 13th edn. Pearson Education, Boston (2009)
8. Cohn, M.: Succeeding with Agile: Software Development using Scrum, 2nd edn. Pearson Education, Boston (2010)
9. Cooper, A., Reimann, R., Cronin, D.: About Face 3: The Essentials of Interaction Design. Wiley Publishing, New York (2007)
10. Dittrich, Y., Rönkkö, K., Eriksson, J., Hansson, C., Lindeberg, O.: Cooperative method development. *Empirical Softw. Eng.* **13**(3), 231–260 (2008)
11. Ferreira, J.: Agile development and UX design: towards understanding work cultures to support integration. In: Bajec, M., Eder, J. (eds.) CAiSE Workshops 2012. LNBP, vol. 112, pp. 608–615. Springer, Heidelberg (2012)
12. Følstad, A., Law, E.L., Hornbæk, K.: Outliers in usability testing: how to treat usability problems found for only one test participant? In: Proceedings of the 7th Nordic Conference on Human-Computer Interaction: Making Sense Through, pp. 257–260 (2012)
13. Fowler, M., Highsmith, J.: The Agile manifesto. *Softw. Dev.* **9**(8), 28–35 (2001)
14. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design science in information systems research. *MIS Quart.* **28**(1), 75–105 (2004)
15. Hudson, W.: User stories don't help users. *Interactions* **20**, 50–53 (2013)
16. Inayat, I., Salim, S.S., Marczak, S., Daneva, M., Shamshirband, S.: A systematic literature review on agile requirements engineering practices and challenges. *Comput. Hum. Behav.* **51**, 915–929 (2015)
17. Jia, Y., Larusdottir, M.K., Cajander, Å.: The usage of usability techniques in scrum projects. In: Winckler, M., Forbrig, P., Bernhaupt, R. (eds.) HCSE 2012. LNCS, vol. 7623, pp. 331–341. Springer, Heidelberg (2012)

18. Jurca, G., Hellmann, T.D., Maurer, F.: Integrating Agile and user-centered design: a systematic mapping and review of evaluation and validation studies of Agile-UX. In: Agile Conference (AGILE), pp. 24–32. IEEE, Orlando (2014)
19. Juristo, N., Moreno, A.M., Sanchez-Segura, M.I.: Guidelines for eliciting usability functionalities. *IEEE Trans. Softw. Eng.* **33**(11), 744–758 (2007)
20. Moreno, A.M., Yagüe, A.: Agile user stories enriched with usability. In: Wohlin, C. (ed.) XP 2012. LNBP, vol. 111, pp. 168–176. Springer, Heidelberg (2012)
21. Nazzaro, W., Suscheck, C.: New to user stories? (2010). <https://www.scrumalliance.org/community/articles/2010/april/new-to-user-stories>
22. Nielsen, J.: 10 Usability Heuristics for User Interface Design (1995). <http://www.nngroup.com/articles/ten-usability-heuristics/>
23. Nielsen, J.: Agile development projects and usability (2008). <https://www.nngroup.com/articles/agile-development-and-usability/>
24. Norman, D.: Ad-hoc personas & empathetic focus. *Jnd. org.* (2004). http://www.jnd.org/dn.mss/personas_empath.html
25. Ramesh, B., Cao, L., Baskerville, R.: Agile requirements engineering practices and challenges: an empirical study. *Inf. Syst. J.* **20**(5), 449–480 (2010)
26. Rodríguez, P., Yagüe, A., Alarcón, P.P., Garbajosa, J.: Some findings concerning requirements in Agile methodologies. In: Bomarius, F., Oivo, M., Jaring, P., Abrahamsson, P. (eds.) Product-Focused Software Process Improvement, pp. 171–184. Springer, Heidelberg (2009)
27. Schwaber, K., Sutherland, J.: The scrum guide. Scrum Alliance (2011)
28. Seffah, A., Gulliksen, J., Desmarais, M.C.: Human-Centered Software Engineering - Integrating Usability in the Software Development Lifecycle, vol. 8. Springer, Netherlands (2005)
29. Sohaib, O., Khan, K.: Incorporating discount usability in extreme programming. *Int. J. Softw. Eng. Appl.* **5**, 51–62 (2011)
30. Sy, D.: Adapting usability investigations for Agile user-centered design. *J. Usability Stud.* **2**, 112–132 (2007)



Increasing the UX maturity level of clients: A study of best practices in an agile environment

Elaine, E.G. Buis^{*}, Simone, S.R. Ashby, Kristel, K.P.A. Kouwenberg

Tilburg Center for Cognition and Communication, Tilburg University, Warandelaan 2, 5037 AB Tilburg, The Netherlands



ARTICLE INFO

Keywords:
 Client-UX relationship
 Case study
 UX maturity
 Agile UX
 UX practitioners
 UX strategy
 Best practices

ABSTRACT

Context: While multiple studies have attempted to define and measure User Experience (UX) Maturity — i.e., how familiar organizations are with UX concepts or strategies— more practice-based insight is needed to examine how UX practitioners maneuver in their relationships with low UX Maturity organizations and help these clients become more ‘UX Mature’.

Objective: This study evaluates how UX practitioners work with low UX Maturity clients, what obstacles they face, and how they cope with these obstacles. From these insights, a set of best practices are identified for UX practitioners who work with low UX Maturity clients and wish to increase their clients’ UX maturity in an agile environment.

Method: These best practices were collected in the form of case studies, involving a total of 20 case studies based on interviews with 22 UX practitioners. The case studies reflect on past projects that were conducted for clients with a low UX Maturity level. Data was obtained through semi-structured interviews and analyzed using a grounded theory approach combined with elements of a thematic analysis.

Results: The results help to identify frequently experienced obstacles in working with low UX Maturity organizations, as well as six best practices for increasing the UX Maturity of these clients.

Conclusions: The study results demonstrate that UX practitioners indeed fulfill a significant role in overcoming organizational UX boundaries. A Low UX Maturity Best Practice model was developed, which summarizes how UX practitioners can optimize their impact in working with low UX Maturity clients, while simultaneously contributing to a more user-centered focus on the part of their clients.

1. Introduction

To create a successful digital product, offering a wide variety of functionalities is no longer solely sufficient; several quality aspects should also be implemented [19]. One of the most important aspects is User Experience (UX), relating to the experience that a user has with the product. Reported benefits of UX design entail: products that are better suited to the target audience [21], and motivating the future use of products [14], thereby also contributing to higher levels of user satisfaction [2]. Providing a positive UX in products is crucial for agile product development organizations to grow, increase efficiency, and gain market shares [8,26,39], with a growing number of organizations reaching out to UX practitioners to aid them in optimizing their products [37,38].

Although clients would generally like to increase the impact of UX practitioners’ digital designs, they are often unable to allocate resources

(e.g., time and access to user data) to provide the optimal level of support [28,7]. This inability is typically caused by misunderstandings or a lack of awareness on the part of the client as to what UX design as a process entails [39,43], and how it could best be implemented in their existing agile environment [9,17]. Therefore, many clients struggle to prioritize integrating the concept of UX design into their agile development processes and corporate culture [12,40]. Research suggests that UX practitioners are typically seen as creators of visually aesthetic user interfaces (UIs) by organizations, indicating that the role of such designers in specifying crucial product requirements may be overlooked [16]. As a result, several organizations merely adopt UX design as a strategy to differentiate their products’ visual appeal from those of their competitors [17].

Ideally, clients should operate with a user-centered mindset, i.e., be intrinsically motivated to provide positive user experiences [33]. As mentioned by Sauro, Johnson and Meenan [39], organizations need to

* Corresponding author.

E-mail address: elainebuis@gmail.com (E.E.G. Buis).

understand and commit to improving the UX on all levels, since without users, products and services lack any worth. Briefly considering UX design as an isolated component of the agile product development process is, therefore, not solely sufficient [9]. To achieve lasting impact, a product's UX should be considered early, continuously, and thoroughly throughout the entire design process and supporting organization [9,33,39]. Seeing UX design as part of the agile development process, where UX designers and developers align their working strategies, could not only enable a smoother workflow, but is also a key indicator for optimally meeting user needs [9]. A constitutional shift in corporate identity and processes is often necessary to commit entirely to the user-centered mindset that defines organizations with a positive UX reputation [5,28].

For UX practitioners who begin working with a client who is still in the process of adopting a user-centered mindset, some challenges and growing pains could arise [28]. Examples of these pains found in related work include the execution of UX activities at too late a stage to inform essential design decisions [29], as well as curtailed access to user needs and preferences [24]. This means that UX practitioners and their clients can take decidedly different stances in the design process [24].

In order to align mutual expectations, UX practitioners have to decide how to approach the client, with these early interactions serving as key determinants in the client's shift to becoming more user-centered [28]. Thus, it is largely up to UX practitioners to evangelize the importance of UX design with their clients as an integrated part of the agile methodology. In this process, a lack of resources allocated to UX design processes [27] and feeling the pressure to 'sell' UX design as a method to clients to obtain resources [4] are frequently experienced challenges reported by UX practitioners. While multiple researchers have built theories and models to support UX practitioners in implementing UX design correctly (e.g., [28,38,36,31]), the current study focuses on best practices for supporting a client's growth in adopting UX design (i.e., fostering the client's UX Maturity).

Chapman and Plewes [5] suggest that identifying a client's familiarity with UX concepts and strategies – hereafter referred to as the client's UX Maturity level – could be key in tailoring UX approaches and realizing good design outcomes. While past studies have focused on finding methods to determine a client's UX Maturity level [38,5,36], or finding correlations between UX Maturity and product success [10], more light has yet to be shed on the definition of best practices that UX practitioners use to adapt, create impact, and help their client grow more 'UX Mature'. Even though some studies have proposed solutions for isolated challenges, e.g., using tailored lean or agile design processes to cope with scarce resources [11] or knowledge-building sessions with the client to increase UX acceptance [28], there is little research on how practitioners maneuver around low UX Maturity obstacles in practice.

To gain insight in how UX practitioners accommodate to low client UX Maturity levels for yielding good outcomes for the project and client relationship and UX Maturity, this study will build on the UX Maturity model of Chapman and Plewes [5]. This model will be discussed in the next section. Subsequently, new insights into the complex relation of low maturity clients with UX design will be provided by examining a set of case studies of projects carried out at different organizations. The challenges and potential solutions of working at these organizations will be discussed in Section 3 (Results). In Section 4 (Discussion), a set of best practices is presented, which emerged from a synthesis of this study's results. These insights will help test the validity of the UX Maturity model of Chapman and Plewes [5]. It is hoped that the findings of this study will contribute to a wider understanding of the UX Maturity landscape, and that this collection of best practices may support UX practitioners as they work with clients to appreciate the added value of UX design as an integrated part of the agile development process, potentially leading to clients becoming more accepting of – and familiar with – UX design.

2. Method

2.1. Design

For this research, 20 case studies were collected and examined, involving design projects performed by agile UX practitioners for low UX Maturity clients. Case studies were chosen as a method for the data collection because they provide a holistic perspective on real-life events and the process leading to specific decisions and results [45], thus helping to capture best practices and their effects on the overall design process. Another reason for a multiple case study approach lies in the method's appropriateness for exploring theories and models within an authentic, real-life context [30,42]. This characteristic is fundamental in studying knowledge fields that have not attracted much attention, as in the case of the current study [1]. Case study guidelines were adopted based on [45] to ensure the reliability of this approach.

Data was collected using semi-structured interviews with UX practitioners working with agile strategies and methods. Semi-structured interviews were chosen because of their appropriateness for capturing new ideas that are brought up during the interview, the possibility to explore topics relevant to the interviewee more deeply and thus also their ability to allow probing follow-up questions [43]. A recent design project at a low UX Maturity client was discussed in these interviews. The case study proposition can be divided into two areas: (1) how the project went (problem statement, process, and project outcome) and (2) how the UX practitioners accommodated to the client's UX Maturity level to still yield optimal outcomes for both the agile product development project and the client relationship.

2.2. UX maturity model

Several models offer a basis for assessing which UX Maturity stage a client is currently at [5,31,39]. What unites these models is the assumption that a client can only implement UX adoption changes that are appropriate for its current UX Maturity level. However, the model by Chapman and Plewes' (see Fig. 1) gives concrete key indicators that enhance the ability to define an organization's UX Maturity [5]. In the first stage of this UX Maturity model, there is a complete absence of UX design in the organization, whereas in the fifth and final stage, UX design can be seen as a part of the organization's identity and corporate objectives.

Chapman and Plewes further define six key indicators (see Table 2) that help pinpoint an organization's Maturity Stage [5]. These indicators – which were found relevant in a series of case studies by Fraser and Plewes [10] – provide the basis for using this Maturity model in the current study. Therefore, an interview protocol for gathering case study data was developed based on the six key indicators of the UX model by Chapman and Plewes [5]. This list consisted of questions about the client, challenge, design process, key indicators and how they were experienced in the project, client relationship, and project outcome

It is important to note that this UX Maturity model is a generalization; in practice, clients often either present a combination of characteristics from different stages, or are in the process of transitioning from one stage to the next. Chapman and Plewes, therefore, characterize their model as one that provides indicators to UX practitioners for applying to their organization, instead of measuring UX Maturity in absolute terms [5].

2.3. Participants

A total of 22 UX practitioners participated in this study. All practitioners worked in the Netherlands and used agile methods and strategies, such as scrum and lean software development. Three UX practitioners worked on the same project together (and were part of the same case study), while the remaining participants discussed separate projects. All practitioners were recruited through professional

Timing of initial UX	Timing of initial UX	Timing of initial UX	Timing of initial UX	Timing of initial UX	Timing of initial UX
None	After coding	Mixed; after coding, before coding, sometimes parallel to coding	Prior to coding	As part of business and market requirements	
Availability of resources	Availability of resources	Availability of resources	Availability of resources	Availability of resources	Availability of resources
None	Visual design	Visual, interaction, research, lower management, and upper-level management	Visual, interaction, research, lower-level management	Visual, interaction, research, lower-level management	Visual, interaction, research, management executive
User involvement	User involvement	User involvement	User involvement	User involvement	User involvement
None	Limited to user testing design/ functionality	Limited to user testing design/ functionality	Correct in all stages of the design process	Correct in all stages of the design process	
Leadership and culture	Leadership and culture	Leadership and culture	Leadership and culture	Leadership and culture	Leadership and culture
None	None	No clear owner, shared across functions	Clear owner in upper management	Clear owner, present at the executive level	
Corporate processes	Corporate processes	Corporate processes	Corporate processes	Corporate processes	Corporate processes
None	None	None or beginning to implement	Successful implementation, Reputation	UX goals are all linked to corporate goals.	
Design thinking	Design thinking	Design thinking	Design thinking	Design thinking	Design thinking
None	None	Not standard	Applied throughout the organization	Applied everywhere from UX to executive level	
Stage 1 Beginning	Stage 2 Awareness	Stage 3 Adopting	Stage 4 Realizing	Stage 5 Exceptional	

Fig. 1. The UX maturity model [5].

networking sites. Table 1 shows their roles, years of experience, size of the client organization, the client's industry, and the nature of the design project. Practitioners all had an educational background in communication, design, or media, with 20 practitioners (90.9%) having obtained a higher degree. The sample of UX practitioners consisted of those working freelance ($n = 4$), those who acted as consultants for UX and other design firms ($n = 7$), and those whose employment was fixed ($n = 11$). At the end of the study, all pseudonymized insights and findings were shared with the UX practitioners.

The client organizations ranged in size from a minimum of 50 employees to clients with more than 20,000 employees. Most clients presented key indicators from two stages of the UX Maturity model. In such cases, the UX practitioners assessed their client's UX Maturity level as being between stages two and three ($n = 15$) at the start of the project, whereupon these same participants reported witnessing a noticeable increase in UX Maturity on the part of their client over the course of the projects discussed herein. Low UX maturity manifested itself primarily in the presence of few resources, a lack of understanding about the

process, and a hesitation to fully commit to UX design from the start of the project.

2.4. Data collection

This study received ethical clearance for data collection in November 2020 by the Ethics Committee of the Tilburg School of Humanities and Digital Sciences. Before taking part, all UX practitioners were asked to fill in an online informed consent form. The participants were also made aware of their rights and were ensured that all data could not be traced back to specific UX practitioners or clients.

Case study data were obtained through semi-structured interviews, which lasted 45 – 60 min and were audio-recorded. Data collection happened online via video conferences due to Coronavirus measures. At the start of the interview, the practitioners were asked to provide information about the client organization, their corporate culture, industry, and the design challenge. Secondly, the design method that the UX practitioners used was discussed. Specific attention was paid to the

Table 1
Practitioner and project details.

Code	Role	Contract	Exp. years	Size client org.	Industry client	Design project
UD_1	UX designer	Consultant	2 years	<100	Payroll services	HR support tool
UD_2	UX/UI designer	Freelance	4 years	<1000	Household goods	New website and webshop
UD_3	UX designer	Fixed	7 years	<2500	Network operator	Internal processing tool
UD_4	UX designer	Fixed	10 years	>10,000	Technical B2B	Customer support handbook
UD_5	UX designer	Freelance	7 years	>2000	Travel industry	Digitalize processes in an app
USP_6	UX specialist	Fixed	10 years	<2500	Furniture	Optimize UX for a webshop
UD_7	UX designer	Freelance	2 years	<100	Housing provider	New website and web app
SD_8	Service designer	Consultant	7 years	<3500	Education	Flexibility vision and toolkit
UD_9	UX designer	Fixed	2 years	<300	Agriculture	Software management system
UD_10	UX designer	Consultant	5 years	>10,000	B2B services	B2B communication portal
UD_11	UX/UI designer	Freelance	4 years	>100	Investments	Calculating tool
UR_12	UX researcher	Consultant	5 years	<50	Software management	Real estate management software
UD_13	UX designer	Fixed	7 years	<1500	Insurance & undertaking	Web application funeral services
UST_14	UX strategist	Consultant	12 years	<6000	Construction group	Innovation platform
UD_15	UX designer	Consultant	5 years	>20,000	Finance	Optimize communication process
UL_16	UX lead	Fixed	6 years	>300	Transportation	Optimize UX of an app
UD_17	UX designer	Fixed	3 years	<2500	Network operator	Planning application
UST_18	UX strategist	Consultant	7 years	<3500	Insurance	User-validate a concept
UD_19	UX designer	Fixed	5 years	<250	Education software	New software system
UST_20	UX strategist	Fixed	9 years	>10,000	Technical B2B	UX consistency platform
UD_21	UX designer	Fixed	8 years	>10,000	Technical B2B	UX consistency platform
UA_22	UX architect	Fixed	8 years	>10,000	Technical B2B	UX consistency platform

six key indicators of UX Maturity, how they were perceived, and how UX practitioners ensured that their impact was optimal. The participants were probed to give best practices that they used to accommodate to challenges by naming examples of the literature (e.g., ‘in coping with limited resources, the literature suggests using secondary sources, did this also occur in this process?’).

After the discussion of the key indicators, the project’s outcome was examined and specifically if the UX practitioners were satisfied with the outcome. Subsequently, the client’s relation with UX design before, during, and after the project was discussed. The UX practitioners were asked if they could pinpoint in which stage the client was before and after the project, and the improvement of which of the six indicators would be most crucial to move to an ascending stage. To probe the participants’ recollections and aid in recalling the different thoughts and emotions that arose during the project, interviewees were asked to share and discuss artifacts from the design process (prototypes, the final product, a presentation given to management, etc.). In the case of the three UX practitioners who collaborated on the same project, thus reflecting one case study, additional questions were asked to explore possible differences in opinion and perception with respect to the project.

2.5. Data analyses

In the analysis of the pseudonymized data, the Grounded Theory (GT) approach by Strauss and Corbin [40] was used to study the events that showcased best practices used by UX practitioners to adapt to their clients and help them grow more ‘UX Mature’. Data gathering and analysis were carried out simultaneously until theoretical saturation was attained. Interviews were partially transcribed and supplemented with notes taken during the interviews. Furthermore, infographics were made per case study (see Fig. 2 for an example) to provide an overview of each UX project, to synthesize results while keeping individual differences in mind.

The infographics showed the aforementioned case study proposition and provided a quick outlook of the discussed project, highlighting the challenge, process, UX Maturity challenges and used best practices, the outcome, and the client relationship. All infographics were reviewed by the associated UX practitioner as a final check on information presented.

In line with the GT approach by Strauss and Corbin [39], data analysis followed a process that started with coding, in which meaningful excerpts from the interview transcripts were deductively coded

into UX Maturity key indicators. During the coding phase, a representative quote – e.g., ‘I was just trying to constantly prioritize activities to be as effective as possible’ (UL_16) – was associated with each code (in this case, ‘time pressure’). Next, a set of themes was identified describing the observed codes. These included: ‘time pressure’, ‘creating positive PR for UX’, and ‘scoping and prioritizing work activities’. The next step in the GT-approach was to apply axial coding. Here, the key indicators from the UX Maturity model (e.g., ‘initial UX involvement’ and ‘leadership and culture’) were matched to the coded data. Here, ‘time pressure’ was, for example, organized under the axial code ‘available resources’.

The final phase of our GT approach used selective coding to find overarching themes in the axial coding data. The code ‘available resources’ was placed under the main theme ‘Early and continuous commitment to UX’. The three themes that emerged from the selective coding phase of the data analysis were: ‘Comprehension of the meaning and value of UX’, ‘Early and continuous commitment to UX’, and ‘Openness to user-centered design processes’. These themes and related data will be discussed in depth in the results section.

Finally, information that was coded during the GT approach as being a potential solution for the challenges described by participating practitioners was also thematically organized using the key indicators of the UX Maturity model. The approach of Xu and Zammit [46] for thematically mapping multiple data sources in a practitioner analysis was adopted. The aforementioned challenges per key indicator from the UX Maturity model [5] were bundled with successful approaches that the participating practitioners said employed to improve their impact. Affinity diagramming was used to cluster all of the themes that emerged per key indicator and find patterns. An example of a solution-based comment identified as pertaining to ‘leadership and culture’ was: ‘In this project, it felt like there were three captains on one ship! I felt like I had to give a lot of trainings to define a clear UX strategy’ (UD_4). An example of a solution-based comment, identified as being related to ‘initial UX involvement’ was: ‘At one point I was trying to save time by planning feedback “clean-up UI” sprints together with developers, and thus simultaneously educating the developers and doing my work’ (UD_9). The six key indicators of the UX Maturity model thus served as a base for six emerging best practices. For example, the accompanying best practice for the indicator ‘leadership and culture’ is ‘Lead by example’. These best practices and the related Low UX Maturity Best Practice Model are outlined in detail in the Discussion section.

Table 2

Key UX maturity indicators with their definitions from [5] and best practices from literature.

Key indicator [5]	Definition from [5]	Best practices
UX leadership and culture	Measures how the value and urgency of UX design are perceived by management and the organization in its entirety.	<ul style="list-style-type: none"> ○ Education and trust-building in the form of trainings and workshops, guides, and frameworks [28]. ○ Starting a fun, open-invitation project to stimulate UX participation [7]. ○ Quantifying the added value of good UX [23]. ○ Meticulous planning of activities and making them concrete for the client [22]. ○ Parallel design processes in which UX practitioners and developers iterate the design and development of a product separately and simultaneously [41]. ○ Lean/agile design processes that save resources due to short production cycles and removing waste from the design process [11,15,25,32]. ○ Design sprints; time-constrained production cycles [20]. ○ Protopersonas that are assumption-based, and validated through user testing [11, 18]. ○ Second-hand resources to learn more about the user, e.g., social media, existing literature, market research, expert reviews about the users [3,15]. ○ Mapping out the business culture and the role of each of its stakeholders to advocate for the support of UX design in current organizational processes [35]. ○ Let management observe users engaging with their products in real-time to motivate the user-centered mindset [39]. ○ Making all stakeholders aware of the value of design thinking [6]. ○ Merging DT with existing practices; demonstrating the usefulness of DT through external resources (research, speakers, etc.); developing metrics to measure the usefulness of DT; and supporting the development of bottom-up initiatives with training [34].
Timing of initial UX	Measures when UX design is initially introduced within the design process.	
Availability of resources	Measures to what extent organizations have access to UX expertise and resources.	
User involvement	Measures to what extent users are involved in the design process.	
Integration of UX in corporate processes	Can be used to estimate the degree to which UX processes are intertwined with corporate processes.	
Application of design thinking (DT) in corporate processes	Focuses on proper definition of the problem before pursuing solutions, as integrated in all corporate processes.	

3. Results

3.1. Comprehension of the meaning and value of UX

The UX practitioners stated that clients often underestimated the return-on-investment of UX design and misinterpreted its ultimate cost-saving benefits. It was apparent from the perspective of multiple UX practitioners that clients' initial motivation to improve their product's UX design was often more related to staying ahead of the competition than genuinely wanting to provide a more positive experience for users. This conviction resulted in a resistance on the part of clients to fully commit to the user-centered mindset associated with UX Maturity. This was evident among clients favoring visual design over the user interaction, and letting developers make decisions about essential UX elements.

'UX design is still seen too much as just a way of saving money and time, and this reflects the corporate goals.' — UD_15

Fifteen UX practitioners (68.1%) experienced challenges with integrating UX design in corporate processes. These practitioners stated that UX goals are often clear for a product, but not at the corporate level. They furthermore indicated that clients' key performance indicators and mission statements typically did not reflect user-centered goals. This lack of integration also had to do with clients being hesitant about innovation and making substantial changes, as was apparent across multiple case studies. However, some organizations were excited about the innovation, even if they do not fully comprehend it yet. One UX practitioner stated: '*Design thinking is a 'marketing' term that helped to sell UX to management, even if they only understand half of its meaning'* (UD_21). Another UX practitioner stated that she wished that she had '*given the client a quick UX course for a tangible feel of what UX is all about'* (UD_1).

An indicator that is strongly interlinked with the implementation of UX design in corporate goals is the presence of UX leadership and culture. Thirteen UX practitioners (59.1%) felt that the UX leadership and culture were not yet sufficiently in place in the client organization, which was characterized by a lack of clarity regarding project ownership, or excessive micromanagement. One UX practitioner stated that they were required to '*heavily structure the sessions and re-explain the design thinking cycle'* [SD_8]. UX practitioners indicated that clients

regularly either had an unclear grasp of the meaning of UX design, or supplied concepts according to their own insights. This misunderstanding resulted in almost all UX practitioners spending an additional investment of their time educating clients and lobbying for UX approaches and resources.

'My colleagues and I spent 50 – 70% of our time lobbying for UX design.' — UD_21

The UX practitioners who experienced these implementation challenges stated that defining a clear strategy in combination with educating the client could be helpful. Almost all UX practitioners offered their client workshops, e.g., presenting UX concepts in general, design sprints, persona making, and ideation hackathons. Practitioners also discussed the importance of educating management on the risks of launching a product with bad UX design, e.g., revealing how much money could be wasted if the product is not sufficiently user-tested. Examples from previous projects helped in providing evidence of such phenomena (UD_7). One UX practitioner stated that '*quantifying the value of good UX design with usability metrics goes a long way towards 'selling' the client on a more user-centered approach'* (UL_16).

Furthermore, UX practitioners stressed the importance of including multiple employees from throughout the organization to enlarge the supporting base for UX design. In situations that involved unclear product ownership, some UX practitioners suggested keeping the product owner role within the UX team. However, this solution is contingent upon first gaining the client's trust.

3.2. Early and continuous commitment to UX

Most UX practitioners in this study mentioned that UX design did not have a high priority within their client's organization. This lack of commitment translated to UX practitioners often being: involved too late in the development process of new products, unable to influence the establishment of product requirements, and delayed in the execution of key UX research activities for informing essential design decisions. Consequently, assumptions about users' needs tend to have already been made, leaving UX practitioners with little maneuver room to create impact. The case studies showed that clients typically demand tangible results quickly, thereby neglecting the need to plan carefully, form expectations, and evaluate findings in an established manner. One UX



Fig. 2. Process outlook of one of the case study projects.

practitioner stated: '*the client required visual outputs of the idea, since the client's initial vision of UX had to do more with the 'look and feel' of products than really seeing it as a process* (UR_12)'. Another UX practitioner who operated alone was tasked to '*design the Apple among the agricultural computers*' with very limited means (UD_9). The need to make benefits of UX tangible was also felt strongly by one UX practitioner (UST_18). Operating in this manner risks suboptimal design outcomes and late (and expensive) design changes, as was apparent from several case studies. Six UX practitioners (27.3%) stated that they were involved too late in the design process.

'We were hired 'during coding', and immediately we pointed out some UX flaws. If they had continued, it would have cost them a lot of money.' — UD_3

One UX practitioner stated that when management saw the added value of UX, the project was given an '*financial impulse that helped to accelerate a positive outcome of the project*' (UD_4). Insufficiently prioritizing UX design also relates to limited resources being allocated to the UX research and design effort. These resources include time, existing user data, budget, and experienced staff (e.g., experienced UX developers). Eighteen UX practitioners (81.2%) stated that they often experienced time and budget limitations during the projects in focus, resulting in an increased workload. Being appointed too few resources can also result in UX practitioners having to make difficult decisions about which activities they can and cannot fulfill.

'Because of time pressure, I was constantly trying to estimate the risk of my attention not going somewhere.' — UL_16

All interviewed UX practitioners used some form of a lean or agile design process. These methods helped them to save time and bring structure to the design process. The UX practitioners who experienced time constraints as a challenge also used parallel design processes and design sprints to save time. For example, one UX researcher reported starting one week in advance and delivering the findings directly to the development team, thereby enabling the UX researcher to investigate a new topic in the following week (UD_1). In this manner, work was ongoing and did not come to a standstill. Another practitioner described planned feedback sprints and direct cooperation with developers near the end of the design process to clean up the UI (UD_9)]. Multiple UX practitioners also stressed the importance of professional development activities and obtaining new knowledge and skills, thus avoiding outsourcing certain tasks. One stated; '*I just tried to do as much myself as possible, by being present in every meeting and being aware of all relevant information*' (UST_20). To cope with minimal resources, two UX practitioners further described the use of secondary resources, such as competitor designs (UR_12) and personas from past projects (UD_19), to gain a better perspective on the users' needs.

3.3. Openness to user-centered design processes

According to the interview findings, many clients experience difficulties in adopting a user-centered mindset, in which user involvement and design thinking hold a central place. Ten UX practitioners (45.6%) stated that the involvement of users as stakeholders throughout the design process was insufficient. The main reason for inadequate user involvement was the assumption that clients already know their users, resulting in a reluctance to grant UX practitioners direct access to their customers. One UX practitioner stated that '*it was necessary to create a support base for UX, before the client admitted to maybe not fully knowing the client*' (UD_13). Another one stated, '*users should be central to the elements of the design process, not the other way around, which happened with my client*' (UD_17).

In the case of B2B clients, gaining access to users was sometimes even more difficult, as several UX practitioners reported having to obtain the approval of two organizations to gain this access. One UX practitioner stated: '*I wish I had been able to do one more user test before transitioning to*

the final product, we could have been finished so much sooner' (UD_15). Furthermore, in the experience of 10 practitioners, clients demonstrated a strong preference for user testing over user research. All UX practitioners stated that they would prefer the combination of user research and testing to sufficiently map user needs.

'Continuous access to users would have really helped to create a more suitable design.' — UD_1

Another UX practitioner stated: '*I had to educate the client on UX, they asked a lot of perspective throughout the process and the outcomes*' (UR_12). Additionally, 17 UX practitioners (77.3%) stated that design thinking was not being sufficiently implemented. In multiple case studies, design thinking was present within the design team but missing from the corporate perspective. Practitioners stated that while some clients were familiar with the term, the perception is often too solution-focused. One UX practitioner highlighted the underlying reason that causes this insufficient application as: '*Thinking' doesn't translate to 'doing' it yet*' (UD_1). In another interview, it was reported that the client organization went so far as to completely banish design thinking, as the management came to associate this design approach with instability.

'Design Thinking was a forbidden word that we were not allowed to use.' — UD_19

To promote a user-centered mindset, six UX practitioners stressed the importance of being actively present in the client organization. One UX practitioner stated that this helped to '*make the client see that a UX-centered mindset can also contribute to improved business processes for the entire organization*' (UD_7). They achieved this by being approachable, giving workshops, and including people from different departments in the design process when possible. Training staff to implement design thinking in their respective departments was also described as a helpful practice (UD_13). Some UX practitioners spoke about emphasizing the relative ease of including users in the design process. For example, one UX practitioner held a phone interview with a user in front of management to show what kind of questions they would ask (UD_1). Finally, in the case of inadequate user data, three UX practitioners applied a more context-based approach to derive user insights throughout the development of personas (UR_12; UD_9; UD_19). These UX practitioners placed a greater emphasis on the context in which the product was used.

4. Discussion

This paper focused on the use of the key indicators of the UX Maturity model from Chapman and Plewes [5] to define best practices for UX practitioners working with Low UX Maturity clients in agile environments. These best practices are presented in the following sections so that UX practitioners can use them as a resource for adapting to projects that involve low UX Maturity clients, while contributing to the establishment of a user-centered organization.

The best practices and accompanying recommendations are also summarized in the Low UX Maturity Best Practice model (see Fig. 3). The left side of this model focuses on the UX practitioner's optimal impact in a constrained environment: '*Adapt to optimize impact in every stage of the design process*' (paragraph 4.1). The right side of this model shows additional best practices that UX practitioners might use to sufficiently guide their client in achieving a more user-centered mindset: '*Contribute to the establishment of a user-centered organization*' (paragraph 4.2).

4.1. Adapt to optimize impact in every stage of the design process

Most UX practitioners in this study experienced a late investment of UX strategy, inadequate resources, or limited user involvement throughout the design process. The resulting challenges are similar to those stated by Chapman and Plewes, for example, concerning limited resources, no standard design process, and involving users in the design

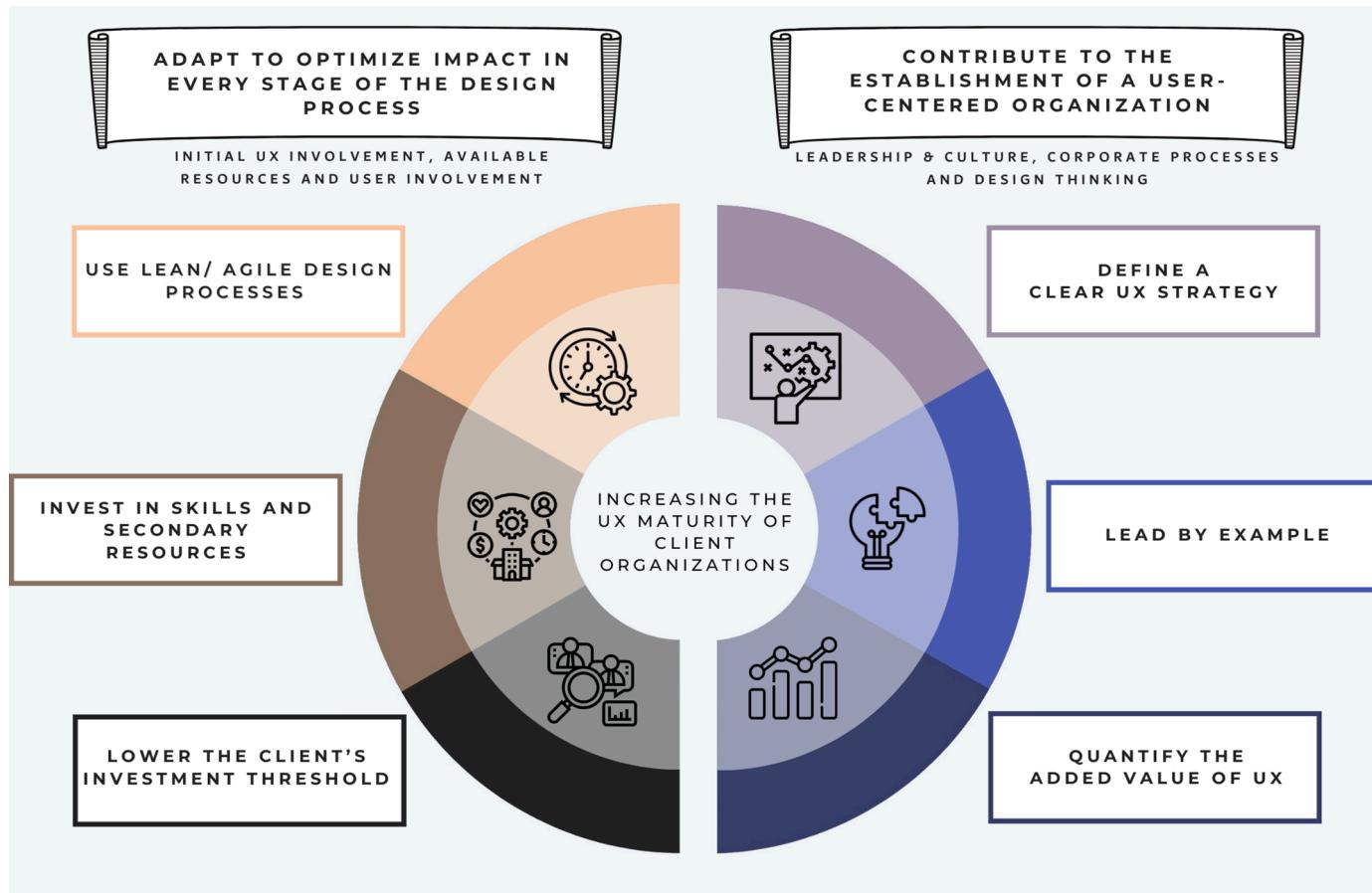


Fig. 3. The low UX maturity best practice model.

process either too late or not at all [5]. This practice is also described by [13]. In the experience of multiple UX practitioners, clients often wanted results quickly and rushed into coding due to the high cost of keeping developers idle. Interview data showed that, in practice, user testing is typically preferred over user research, a finding echoed by [17]. Even though, according to some practitioners, this could potentially lead to faulty personas and ensuing design flaws. Some UX practitioners argued that this preference for user testing stemmed from the client's assumption that they already know the user's needs at the start of the design process.

4.1.1. Use lean/ agile design processes together with parallel processes

UX practitioners in the case studies often used elements of lean or agile design processes, as well as design sprints and parallel processes, for greater efficiency. Several UX practitioners had arrangements with development teams, in which they performed their work activities ahead of time and subsequently provided the results to the development team. The use of these methods is advocated in design processes proposed in [39,25] and [15] for their time and resource-saving qualities. Furthermore, as multiple UX practitioners reported, appropriate design is also about making difficult choices. Sometimes a practitioner had to choose between design elements, thus having to try to correctly estimate the risk of UX design not being fully implemented. In such cases, some UX practitioners suggested the scoping of work activities and the use of a decision matrix to assess risks.

4.1.2. Invest in skills and secondary resources

Multiple case studies show that acquiring new knowledge and skills can help UX practitioners improve their response to certain situations. Some UX practitioners took coding courses to improve communication

with the development team and promote a shared understanding. The interview data showed that this type of knowledge could also be obtained by working closely with other departments and observing what they do and how they operate. Several UX practitioners in the case studies, for example, joined marketing meetings to see how they viewed the user's needs and how they communicated with users.

Several UX practitioners highlighted the need to understand how a client operates and how work processes flow, especially if a client operates in a very specialized work field. Findings in [35] and [12] also underscore the importance of understanding the organizational culture and each stakeholder's role within this culture. Interestingly, none of the UX practitioners interviewed made use of the assumption-based proto-personas presented by [11], instead preferring research-based personas, or, in cases with no access to users, personas that place a user in a context. In the latter case, practitioners gave special prominence to the context of use. A PACT-analysis (People, Activities, Contexts, and Technologies) aided such practitioners in creating these context-based personas.

Another UX practitioner utilized the competition's resources (e.g., website articles) and approach. Other secondary resources used in the case studies included secondary statistics (from the Dutch central bureau of statistics), existing market research with the target user group, and personas from previous projects. These findings align with [3] and [15]. These researchers also advocate the use of secondary resources to gain a better understanding of users in the case of limited resources.

4.1.3. Lower the client's investment threshold

Another best practice to optimize working with low UX Maturity clients is to lower their investment threshold. Multiple UX practitioners did this by concretizing work activities. The use of feasibility and

impact/effort matrices can provide a solution here. These matrices helped several UX practitioners to concretize their work activities and the ensuing design possibilities for the client.

Some UX practitioners stated that since matrices show the result-driven characteristics of UX design, the concretization of options helps to reach a compromise with the client when negotiating strategy. Moreover, UX practitioners used MVPs and low-fidelity prototypes to lobby for more resources. Several practitioners presented a visual example of the product in its most minimal form to persuade the client to choose alternative designs or additional features.

To lower the client's investment threshold, showing how easily user involvement can be accessed can also be a helpful practice. One UX practitioner accomplished this by holding a phone interview with a user in front of management, a best practice also found by [40]. Another UX practitioner showed that feedback from three users could already make a difference and help avoid design mistakes. Diminishing the reluctance to involve users can also help to shatter the illusion that management '*already knows their users best*'. One UX practitioner used the most critical quotes from a user evaluation in a management presentation to show that the product was not experienced in the manner that had been assumed.

4.2. Contribute to the establishment of a user-centered organization

Challenges with UX leadership and culture, the application of design thinking, and the implementation of UX design in corporate processes in the case studies could often be traced back to clients misunderstanding the added value of user-centered design. These findings echo those of [39] and [44]. Due to this misunderstanding, several UX practitioners stated that clients often allocated few resources towards certain aspects of a design process, such as user testing. In the case studies, clients also struggled with UX product ownership during the design process, as well as integrating design thinking in existing processes. Work by [5] and [28] shows that low UX Maturity clients often focus mainly on the visual aspects of UX, a finding that was also evident in the results of this study. The results show that UX practitioners who built trust and guided the client well enabled clients to grow on the UX Maturity scale. For example, one UX practitioner was only hired for three months to improve a digital product's UI. However, after two years, the entire application had been rebuilt through a user-centered design process, and the client had hired two additional UX practitioners.

4.2.1. Define a clear UX strategy

Ideally, most UX practitioners in this study wanted a UX strategist to be added to the project, but in reality, it was mostly up to the practitioners to structure the process, due to limited resources. The significance of a concrete and structured strategy when working with low UX Maturity clients was apparent from multiple case studies. Several UX practitioners described the practice of starting with a kick-off workshop to gauge the client's attitude towards UX concepts and approaches. In this way, an indication of the client's familiarity and openness to UX processes can be established, which can help tailor the UX approach. Two UX practitioners even described using a Design Maturity scale to tailor their UX approach; this scale was very similar to the UX Maturity model used in this study. These UX practitioners used the kick-off workshop to assess this Design Maturity scale and tailor the education and strategy accordingly.

The findings show that demonstrating how UX approaches align with existing corporate processes can be essential. In defining a UX design strategy, mapping out the organization's processes to find bottlenecks could show where and how UX practitioners are of particular value. Multiple UX practitioners used this practice to define growth and improvement opportunities, for example, to create greater efficiency in the production process. A UX process should always be transparent to the client, which some UX practitioners achieved by visualizing the design process steps and project roles in a visual roadmap, or by

mapping all design components in a design system. A design system combines all design components, such as style guides and rules, and thus serves as a guide on how an organization conducts its design processes.

4.2.2. Lead by example

To help clients grow more 'UX Mature', it is necessary to structurally guide them along the way. The findings of the case studies indicated that sessions should be structured and concepts should be made concrete. Concepts such as design sprints and MVPs are often very new and vague for low UX Maturity clients, as was apparent from both the literature [5] and the case studies. Therefore, it is relatively easy for clients to form their own interpretations of concepts and for the practitioner and the client to then communicate at cross purposes. Creating common ground and a shared understanding of each other's expectations and roles is crucial to create a broad UX support base.

Furthermore, multiple UX practitioners stressed the importance of education to create this common ground. As shown in the study by MacDonald [28], education can help foster awareness of correct ways to implement UX design. A client who is very new to UX could initially be educated in UX concepts and techniques. Simultaneously, the UX practitioner could inform the client of other possibilities along the way, such as the accessibility and consistency of features. Almost all UX practitioners interviewed for this study use workshops to concretize UX concepts and techniques for the client.

Additionally, several UX practitioners highlighted the importance of being present and approachable in the client organization. Being approachable enables employees to experience a low threshold to collaborate, learn more about UX, or ask for a UX practitioner's perspective on their work. To make UX design widely accessible, UX practitioners could be present in relevant meetings and give themes for design sprints or feedback sessions. The suggestion to be approachable is also in line with the recommendation by Ede and Dworman [7] to start with an open and fun UX project, in which anyone from within the organization can participate. One UX practitioner mentioned various low-fidelity prototypes for innovation platforms that employees could design themselves. In these projects, employees can directly apply design thinking and learn the correct techniques. Involving employees in such activities, as well as the UX project itself, could contribute to a shared design thinking mindset, as it offers first-hand experience to employees working with this method.

Another way to create a widely supported user-centered mindset in large organizations is to give training in design thinking to heads of departments, who then will further educate their staff and thus implement design thinking in multiple departments. One UX practitioner who worked for a large organization had experienced this practice as being very helpful. This suggestion of phased design thinking implementation is in line with the results of [6] and [34], who stress the importance of making every stakeholder aware of the usefulness of design thinking.

4.2.3. Quantify the added value of good UX design

A frequently heard challenge from the case studies was that UX practitioners felt the need to 'sell UX', a finding that is in line with [4]. Quantifying the value of UX design could help mitigate management distrust. Some UX practitioners explained that qualitatively describing user experiences often felt subjective to clients and enabled them to dismiss the data. However, when metrics are added, the perceived value of UX becomes more concrete and evidence-based. These findings are in line with Lachner et al. [23], who argue that metrics help demonstrate how UX methods are tied to positive user satisfaction levels.

Applying usability metrics or the results of A/B tests are examples of quantifying the added value of UX design. With longer projects, some UX practitioners suggested starting with sharing usability metrics and quantitative data reports with management. In these projects, the initial focus was on providing quantitative evidence, which can then motivate the client to determine the 'why' behind the numbers. Therefore, employing metrics can pave the way for user research in the experience

of these UX practitioners.

Furthermore, results show that metrics can help UX practitioners to convey authority on the subject. Multiple UX practitioners frequently provided this evidence throughout the design process, e.g., giving short demos or using before/after scenarios. This evidence helped to establish valuable PR and a support base for subsequent UX approaches. Last but not least, UX practitioners stated that 'evangelism' for UX design should be shared wider than just with management; it should also be shared with marketing and developer divisions to render user needs more apparent and tangible for these departments.

4.3. Implications and limitations

The study results demonstrated that UX practitioners indeed fulfill a significant role in overcoming organizational UX boundaries. The results appear to support the accuracy of the key indicators of the UX Maturity model of Chapman and Plewes [5]. Subsequently, the data helped to define frequently occurring obstacles and practical solutions that UX practitioners can use to overcome these obstacles. The resulting Low UX Maturity Best Practice model can aid UX practitioners in implementing the corporate and cultural changes necessary to increase agile UX design's overall impact, while simultaneously contributing to the establishment of a user-centered organization. With this study, we demonstrated that while agile practices are often an integral part of project development within organizations, their use within projects can also be essential in increasing a client's UX Maturity. For example, agile UX practices were strategically applied by the participating UX practitioners to save time and increase the client's awareness of the benefits of applying agile methods to increase efficiencies.

By collecting best practices that UX practitioners use in low UX Maturity environments, a new and practical insight into the relationship between UX practitioners and their clients was given. In a previous study regarding good UX implementation in organizations, MacDonald [28] observed that UX practitioners are often responsible for guiding their clients in implementing UX design. This responsibility was also evident in the current study; it appears that the UX practitioners' guiding role is even larger than may have initially been assumed in the research literature. As one UX practitioner stated, 50 – 70% of their time is spent lobbying for the value of UX to create a support base. Maintaining positive relationships with low UX Maturity clients primarily revolves around UX practitioners building trust and guiding clients in the organizational change necessary to become more user-centered.

This study also gave witness to various areas of friction that UX practitioners experienced during these projects. These problems can be attributed primarily to UX design being a relatively new area of expertise, especially for clients unfamiliar with its methods and concepts. This novelty naturally results in a hesitance on the part of such clients to embrace UX design and its concomitant constitutional shift in the early stages of UX adoption. As multiple UX practitioners indicated, education and trust-building are a logical and integral part of this process and should be considered as such. As one UX practitioner stated: '*You have to make the client fall in love with the problem, not with the solution*' (UR_12). UX practitioners could employ their UX and design thinking skills to meet the challenge of creating an appropriate UX adoption strategy for these clients. The Low UX Maturity Best Practice model can aid UX practitioners in making this process more efficient.

The results also contribute to a clearer understanding of the UX Maturity domain. The key indicators of the UX Maturity model of Chapman and Plewes [5] were found valid. Obstacles per indicator were recognized, as well as the acknowledgment of Stage five as the ideal scenario for organizations. Moreover, it was relatively easy for UX practitioners to pinpoint the client's UX Maturity level within the model using the key indicators. One could argue that since all clients indicated signs of multiple UX Maturity stages, the model's scaling should be adjusted to make these stages more distinct. However, as several UX practitioners argued, most clients were in the process of going from one

stage to another. It is important to emphasize that Chapman and Plewes [5] did not intend for their model to be used as a way of measuring UX Maturity in absolute terms, but rather as key indicators for enabling UX practitioners to assess their clients' maturity levels.

Most clients showed signs of maturity at Stages 2 and 3. Maturity Stage 3 was the most frequently discussed stage, which is interesting, as Chapman and Plewes [5] state that this is the most crucial stage to overcome to become 'UX Mature'. The significance of this stage can be attributed to its now-or-never elements; organizations have to decide if they want to fully commit to UX design or if they want to revert to their previous mode of achieving designs. Therefore, appropriate UX designing guidance is especially crucial at this stage.

Another implication of the findings is that the organization's size does not necessarily correlate with a different UX implementation challenge. UX practitioners who worked for clients with either >10,000 employees or <50 employees often faced the same challenges in, for example, establishing a UX culture. Often, these UX practitioners also provided similar best practices, which endorses the Low UX Maturity Best Practice model's appropriateness for organizations of varying sizes.

The generalizability of this study's results is somewhat limited in scope, since only Dutch organizations were sampled. While some of these organizations were multinationals, there could still be a cultural bias. As one interviewee (a UX practitioner working for a multinational firm) stated: '*There can be many differences in how employees approach management. In the Netherlands, there is a lot less organizational hierarchy than in some other cultures*' (UD_4). An approach of UX practitioners educating management, and leading the way, could then be less appropriate due to management being less receptive to shifts in the established hierarchy.

The UX practitioners in this study were encouraged to pinpoint the UX Maturity of their clients themselves using the UX Maturity model as a guide. This could have biased the results, since all observations were aligned with a single theoretical model. Participating designers may have been more invested in their client's UX Maturity growth, since this reflects on their success within the project. As a result, this assessment may suffer from confirmation bias, while failing to account for project challenges that were not as successfully resolved. However, due to privacy concerns, verifying each UX practitioner's assessment of their clients' UX Maturity level was not possible in the current study. Instead of asking the UX practitioners to specify their client's UX Maturity level at the end of the interview, it might have been more beneficial to do so at the beginning to encourage a more critical assessment. Shifting the order of the interview questions in this way might have helped participants to better reflect on how the organization performed at the beginning of the project, when the potential challenges emerged.

Furthermore, it should be noted that the reliability of the data may have been impacted due to confidentiality concerns. Several UX practitioners had signed non-disclosure agreements and were bound by contracts that prevented them from providing specific details about the projects that were discussed. While this did not apply to the sharing of experienced obstacles and best practices, it could cause some data to appear as idiosyncratic due to the need for some UX practitioners to withhold key project details. Nonetheless, the results appear to coincide with earlier studies and reflect an overall willingness on the part of the UX practitioners to contribute essential insights.

5. Conclusion and suggestions for future research

The ultimate goal of this research was to facilitate the work of UX practitioners and help organizations grow towards achieving a more user-centered mindset. While multiple studies have attempted to measure UX Maturity, more practice-based insight was needed to examine how UX practitioners maneuver in their relationships with low UX Maturity organizations and help these clients become more 'UX Mature'. With a growing understanding of the value of UX design, more organizations are expressing an interest in improving the UX design of their

products. However, to achieve this goal, organizations need to commit to the accompanying user-centered mindset.

The results give direction to three possible forms of follow-up research that can potentially contribute to a growing number of organizations becoming more user-centered. Firstly, where this study has focused on finding best practices for UX implementation from the UX practitioner's perspective, future research could focus on the organizational side to create a more balanced model. While UX practitioners often feel responsible for guiding their clients in the UX implementation process [28], this does not necessarily mean there are no possible improvements on the organizational level to encourage openness to UX design. Collecting and defining best practices from the client's perspective could, therefore, be interesting to further study good integration of UX design and optimal cooperation with UX practitioners.

A second suggestion for future research stems from the finding that quantifying the added value of UX may be a crucial tactic in organizations becoming more receptive to UX design, as argued in [23]. However, further research is needed to elaborate upon this quantification strategy. Important questions include: 'Which metrics are most useful at what (maturity) stage?' and 'What metrics are most convincing for which stakeholder group?'. Consequently, it can be worthwhile to explore how an internal PR division might be best deployed to successfully implement UX design within an organization.

Finally, it is recommended that future researchers also sample from a broader spectrum of UX projects to ensure that the solutions identified from Low Maturity organizations in fact lead to success in high UX Maturity organizations. A future study could also take a more critical look at the UX Maturity model that was used in this study and seek ways to refine it.

CRediT authorship contribution statement

Elaine, E.G. Buis: Conceptualization, Methodology, Formal analysis, Investigation, Resources, Data curation, Writing – original draft, Writing – review & editing, Visualization, Project administration. **Simone, S.R. Ashby:** Supervision, Writing – review & editing. **Kristel, K.P.A. Kouwenberg:** Validation, Resources, Writing – review & editing, Data curation, Writing – review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We would like to thank all UX practitioners who took part in this study for their valuable insights and enthusiasm. Furthermore, we would like to extend our gratitude to Tim Scholten, who gave the first practical insights into the topic of UX Maturity, and whose feedback proved pivotal in shaping this study's problem statement.

References

- [1] F. Almeida, J.A. Monteiro, Approaches and principles for UX web experiences: a case study approach, *Int. J. Inf. Technol. Web Eng. (IJITWE)* 12 (2) (2017) 49–65.
- [2] R. Alves, P. Valente, N.J. Nunes, The state of user experience evaluation practice, in: Proceedings of the 8th Nordic Conference on Human-Computer Interaction: Fun, Fast, Foundational, New York, NY, ACM, 2014, pp. 93–102.
- [3] J. Blindheim, A. Wulvik, M. Steinert, Using secondary video material for user observation in the needfinding process for new product development and design, in: DS 84: Proceedings of the DESIGN 2016 14th International Design Conference, Dubrovnik, Croatia, The Design Society, 2016, pp. 1845–1854.
- [4] A. Bruun, M.K. Larusdottir, L. Nielsen, P.A. Nielsen, J.S. Persson, The role of UX professionals in agile development: a case study from industry, in: Proceedings of the 10th Nordic Conference on Human-Computer Interaction, New York, NY, ACM, 2018, pp. 352–363.
- [5] L. Chapman, S. Plewes, A UX maturity model: effective introduction of UX into organizations, in: International Conference of Design, User Experience, and Usability, Cham, Switzerland, Springer Int. Publishing, 2014, pp. 12–22.
- [6] D. Dunne, Implementing design thinking in organizations: an exploratory study, *J. Organ. Des.* 7 (1) (2018) 1–16.
- [7] M. Ede, G. Dworman, Why designers might want to redesign company processes to get to better UX design: a case study, in: Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems, New York, NY, ACM, 2016, pp. 840–848.
- [8] F. Erdős, Economical aspects of UX design and development, in: 10th IEEE International Conference on Cognitive Infocommunications, Piscataway, NJ, IEEE, 2019, pp. 211–214.
- [9] J. Ferreira, H. Sharp, H. Robinson, User experience design and agile development: managing cooperation through articulation work, *Software: Pract. Exp.* 41 (9) (2011) 963–974.
- [10] J. Fraser, S. Plewes, Applications of a UX maturity model to influencing HF best practices in technology centric companies—lessons from Edison, *Procedia Manuf.* 3 (2015) 626–631.
- [11] J. Goethelf, *Lean UX: Applying Lean Principles to Improve User Experience*, O'Reilly Media, Inc, Beijing, China, 2013.
- [12] C.M. Gray, A.L. Toombs, S. Gross, Flow of competence in UX design practice, in: Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, New York, NY, ACM, 2015, pp. 3285–3294.
- [13] M. Gualtieri, Best practices in user experience (UX) design, *Forrester Res.* 1 (2009) 1–17.
- [14] M. Hassenzahl, Experience design: technology for all the right reasons, *Synth. Lect. Hum.-centered Inform.* 3 (1) (2010) 1–95.
- [15] L. Hokkanen, K. Väänänen-Vainio-Mattila, UX work in startups: current practices and future needs, in: International Conference on Agile Software Development, New York, NY, ACM, 2015, pp. 81–92.
- [16] K. Holtzblatt, J. Barr, L. Holtzblatt, Driving user-centered design into IT organizations: is it possible?, in: In CHI'09 Extended Abstracts on Human Factors in Computing Systems, New York, NY, ACM, 2009, pp. 2727–2730.
- [17] M. Isomursu, A. Sirotkin, P. Voltti, M. Halonen, User experience design goes agile in lean transformation—a case study, in: 2012 Agile Conference, Cham, Switzerland, Springer Int. Publishing, 2012, pp. 1–10.
- [18] P. Jain, S. Djamasbi, J. Wyatt, Creating value with proto-research persona development, in: International Conference on Human-Computer Interaction, Cham, Switzerland, Springer Int. Publishing, 2019, pp. 72–82.
- [19] P. Kashfi, R. Feldt, A. Nilsson, Integrating UX principles and practices into software development organizations: a case study of influencing events, *J. Syst. Softw.* 154 (2019) 37–58.
- [20] W.J. Keijzer-Broers, M. de Reuver, Applying agile design sprint methods in action design research: prototyping a health and wellbeing platform, in: International Conference on Design Science Research in Information System and Technology, Cham, Switzerland, Springer Int. Publishing, 2016, pp. 68–80.
- [21] A.E. Krueger, K. Pollmann, N. Fronemann, B. Foucault, Guided user research methods for experience design—a new approach to focus groups and cultural probes, *Multimodal Technol. Interact.* 4 (3) (2020) 43.
- [22] K. Kuusinen, K. Väänänen-Vainio-Mattila, How to make agile UX work more efficient: management and sales perspectives, in: Proceedings of the 7th Nordic Conference on Human-Computer Interaction: Making Sense Through Design, New York, NY, ACM, 2012, pp. 139–148.
- [23] F. Lachner, P. Naeglein, R. Kowalski, M. Spann, A. Butz, Quantified UX: towards a common organizational understanding of user experience, in: Proceedings of the 9th Nordic Conference on Human-computer Interaction, New York, NY, ACM, 2016, pp. 1–10.
- [24] E.L.-C. Law, P. van Schaik, V. Roto, Attitudes towards user experience (UX) measurement, *Int. J. Hum.-Comput. Stud.* 72 (6) (2014) 526–541.
- [25] L.A. Liikanen, H. Kilpiö, L. Svan, M. Hiltunen, Lean UX: the next generation of user-centered agile development?, in: Proceedings of the 8th Nordic Conference on Human-computer Interaction: Fun, Fast, Foundational, New York, NY, ACM, 2014, pp. 1095–1100.
- [26] K.Y. Lin, C.F. Chien, R. Kerh, UNISON framework of data-driven innovation for extracting user experience of product design of wearable devices, *Comput. Ind. Eng.* 99 (2016) 487–502.
- [27] C.M. MacDonald, It takes a village": on UX librarianship and building UX capacity in libraries, *J. Library Adm.* 57 (2) (2017) 194–214.
- [28] C.M. MacDonald, User experience (UX) capacity-building: a conceptual model and research agenda, in: Proceedings of the 2019 on Designing Interactive Systems Conference, New York, NY, ACM, 2019, pp. 187–200.
- [29] V.K. de Meerendré, L. Rukonić, S. Kieffer, Overcoming organizational barriers to the integration of UX methods in software development: a case study, in: International Conference on Human-Computer Interaction, Cham, Switzerland, Springer Int. Publishing, 2019, pp. 263–276.
- [30] C. Meyer, A case in case study methodology, *Field Methods* 13 (4) (2001) 329–352.
- [31] J. Nielsen, *Corporate Usability Maturity: Stages 1-4*, Nielsen Norman Group, 2006.
- [32] G. Nudelman, Lean UX communication strategies for success in large organizations, *Interactions* 25 (5) (2018) 80–82.
- [33] T. Øvad, L.B. Larsen, The prevalence of UX design in agile development processes in industry, in: 2015 Agile Conference, Piscataway, NJ, IEEE, 2015, pp. 40–49.
- [34] I. Rauth, L. Carlgren, M. Elmquist, Making it happen: legitimizing design thinking in large organizations, *Des. Manag.* 9 (1) (2014) 47–60.
- [35] J.A. Rohn, How to organizationally embed UX in your company, *Interactions* 14 (3) (2007) 25–28.

- [36] L. Rukonić, V.K. de Meerendré, S. Kieffer, Measuring UX capability and maturity in organizations, in: International Conference on Human-Computer Interaction, Cham, Switzerland, Springer Int. Publishing, 2019, pp. 346–365.
- [37] D. Satterfield, M. Fabri, User participatory methods for inclusive design and research in autism: a case study in teaching UX design, in: International Conference of Design, User Experience, and Usability, Cham, Switzerland, Springer Int. Publishing, 2017, pp. 186–197.
- [38] J. Sauro, K. Johnson, C. Meenan, From snake-oil to science: measuring UX maturity, in: Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems, New York, NY, ACM, 2017, pp. 1084–1091.
- [39] M. Stone, F. Bentley, B. White, M. Shebanek, Embedding user understanding in the corporate culture: UX research and accessibility at Yahoo, in: Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems, New York, NY, ACM, 2016, pp. 823–832.
- [40] A. Strauss, J. Corbin, Basics of Qualitative Research Techniques, Sage publications, Thousand Oaks, CA, 1998.
- [41] D. Sy, Adapting usability investigations for agile user-centered design, *J. Usabil. Stud.* 2 (3) (2007) 112–132.
- [42] T. Vissak, Recommendations for using the case study method in international business research, *Qual. Rep.* 15 (2) (2010) 370–388.
- [43] C. Westby, A. Burda, Z. Mehta, Asking the right questions in the right ways: strategies for ethnographic interviewing, *ASHA Leader* 8 (8) (2003) 4–17.
- [44] J. Winter, K. Rönkkö, M. Rissanen, Identifying organizational barriers—a case study of usability work when developing software in the automation industry, *J. Syst. Softw.* 88 (2014) 54–73.
- [45] R.K. Yin, Case Study Research: Design and Methods, 3rd ed., Sage Publications, Inc, Newbury Park, CA, 2003.
- [46] W. Xu, K. Zammit, Applying thematic analysis to education: a hybrid approach to interpreting data in practitioner research, *Int. J. Qual. Methods* 19 (2020), 1609406920918810.

The Use of Stories in User Experience Design

Dan Gruen
IBM Research
IBM Corporation

Thyra Rauch
Tivoli Software
IBM Corporation

Sarah Redpath
Lotus Development
IBM Corporation

Stefan Ruettinger
IBM Software Development
IBM Corporation

Stories capture the characteristics of the design space and audience that designers and engineers need to understand to build a complete and useful software experience. A story is a design communication tool that transcends the cultural divides of multidisciplinary teams and intertwines a technology with its user's goals. This article describes how stories are powerful tools in software design, defines the elements that make a compelling story, and presents the use of stories at IBM from the authors' experience. It also explores the benefits at each phase of the design process and how stories evolve throughout the design process.

1. INTRODUCTION

To design a system that will delight, provide value, and “feel right,” designers and engineers need a deep understanding of the people for whom they are designing, their goals and values, the settings in which they live and work, and their activities. Stories capture this understanding in a rich and meaningful way, in a form that the various members of a product team can relate to and use.

The authors of this article thank Maida Eisenberg for her editorial assistance. We also acknowledge others who helped in creating the examples we used: Stephan Feger, Ursula Hart, Marion Behnen, Gary Macomber, and Tina Adams.

Requests for reprints should be sent to Dan Gruen, IBM Research, 1 Rogers Street, Cambridge, MA 02142. E-mail: daniel_gruen@us.ibm.com

Stories have been used successfully by a number of groups at IBM, and their use as an element of the product development process is growing. We represent three different groups who have found story-based methods valuable throughout the product lifecycle, from inspiring the earliest vision, through concept definition, requirements specification, design, prototyping, development, and product introduction. Stories have helped us ground the design of a new system in the users' business context and environment by immersing designers and developers in the situations in which their systems will be used. Stories help designers determine which functions will be useful; how they should be presented; and what integration with other tools, people, and information will be important. Stories can prompt innovation by revealing new opportunities to provide value to end users. Because they are powerful communication and teaching tools, stories can also help multidisciplinary teams work together, and help end users understand and discuss how a system would fit into their lives.

In this article, we describe what we mean by stories and outline the elements needed to build compelling stories. We explain why stories are a powerful artifact in human-computer interaction (HCI) work. We then give examples of how stories have been used by several different groups at IBM and describe how a story evolved during the different stages of a project process. We conclude by discussing issues to consider when using stories as design tools.

2. WHAT ARE STORIES AND WHY ARE THEY POWERFUL?

As the terms are used differently by different practitioners, it is important to define here what we mean by *stories*, and how we see them as differing from many *scenarios*. In general, scenarios focus on developing the right sequence of actions needed to capture and convey an activity, largely as a way of defining requirements. Scenarios around a technological artifact typically focus on the way a system is used to perform a specific task. Scenarios can vary greatly in level of detail, though many do not include detailed descriptions of the people involved in a task, their motivations, values, or goals. Scenarios also often lack the plot development and drama integral to a compelling story.

For example, a scenario showing the sequence of actions and screens used to fill out an online expense report might not give detail about who is using the system, how the expense report task fits into a broader context and goal, or why it is beneficial to use an online system. Scenarios have been used extensively in training (Schank, 1997) and in HCI for communication and requirements definition (Carroll, 1995; Rosson, 1999).

Stories, on the other hand, are very specific. They include fleshed-out characters and settings, dramatic elements, well-formed plotlines, and enough detail to understand the people who will use a system and the value it will bring to their lives (Burroway, 1999; McKee, 1997). It is important to note that the lines between scenarios and stories are by no means firmly drawn, and many practitioners create scenarios that do contain some story elements. Our point is that we have found ele-

ments like plot, drama, and character development—elements optional to a scenario but critical for a compelling story—extremely valuable in our HCI work.

2.1. *Elements of Stories*

A good story has detailed characters with whom the audience can empathize; rich, contextualized settings; goals (what the protagonist is trying to accomplish and why); causality; and obstacles (what problems the protagonist has to overcome to accomplish the goal). Dramatic elements such as time locks (constraints on the time in which the goal must be accomplished) or option locks (constraints on the actions or items that can be used to accomplish the goal) heighten the dramatic impact of the story (Brannigan, 1992; Burroway, 1999; McKee, 1997). It is interesting to note that many of the same elements required to tell a compelling story also are needed to design a useful and compelling system.

Fleshed-out characters. Good stories have fleshed-out characters with details that allow an audience to understand, relate to, and empathize with them. This includes having a sense of their values, fears, weaknesses, and overall goals: knowing what is important to them and what they would like to avoid. Similarly, in design, it is important to understand, in depth, the people who will use and be impacted by a proposed system. Understanding the broader goals and values of these people, and being sensitive to their limitations, is important as design decisions are made. For instance, knowing that a user might have a high school degree and will get only one week of training can guide decisions about the content and structure of the system.

Detailed settings. Compelling stories generally include details of time and place that help the audience situate themselves in the setting in which the story takes place. Similarly, in design, it is important to understand the environment in which a tool or system will be used and the other artifacts and activities with which it must fit. For example, if the designers understand that an environment is crowded, loud, and fast paced, they can address those sensitivities through systems that do not rely on audio feedback yet provide quick access to system information.

Goals and obstacles. The plots of compelling stories typically are based on a conflict or obstacle that the characters overcome to accomplish a goal. Similarly, a designer should have a clear sense of the goals accomplished or the problem solved by using a system. In essence, this is the *raison d'être* of the system: why it is being built in the first place and how the proposed solution is better than others the user could choose.

Motivation. Compelling stories also require a sense that the actions of the characters are clearly motivated. It is critical for the audience to be able to understand the reasons for their behavior. Stories can provide a “motivation walkthrough” similar to a “usability walkthrough,” in which designers ask not only if users can perform a function or use a feature, but why they would choose to do so. For example, the fact that certain users at customer support centers are timed by their management to determine their compensation shows one source of motivation and would likely be a key design factor.

Causality. Stories are more than lists of unrelated events. Events in stories are connected through causal relations, although these relations sometimes only become clear at the end. Similarly, design stories should show the proposed system playing a causal role in furthering the goals of the people who use it, not simply being used in an incidental way. For instance, in researching how users interact with software systems, we have discovered “peripheral tasks” outside of the systems, where users refer to company directories, written notes about solutions to problems, or whiteboard calendars to complete a task. Information from these sources, which are external to the system, and the actions associated with their upkeep, have everything to do with the user’s actions with the system. This is a view where the user’s complete model of the task, not just the system elements of it, is represented with all the relations intact. This view can introduce opportunities to incorporate into the software more of the information and artifacts of the complete task as the user sees it.

Dramatic elements. Dramatic elements and plotlines make stories interesting and emotionally engaging. Dramatic elements heighten the sense that something is at stake and reveal the characters’ core values. Similarly, in system design, anticipating crises and critical situations can lead to systems that are focused and that support essential functions robustly in a variety of situations.

2.2. Our Growing Recognition of the Importance of Drama

The importance of dramatic elements in design stories came as a revelation to us. Initially, our sense was that our stories should focus on describing systems used in typical, unspectacular activities, as a way of emphasizing their value in everyday situations. There was a fear that including crises and other exceptional events would detract from the generalizability of the stories, making it harder for people to map them to their own, presumably more controlled domains. We have learned, however, that including dramatic elements such as time and option locks adds interest to stories and helps focus attention on the essential benefits of the systems they depict. Including such elements leads to stories that involve situations in which the tool will make a critical difference.

For example, a story created around a proposed expertise-location system began by showing the system being used to locate people to work on a sports Web site that would let viewers monitor the positions of athletes during an upcoming triathlon. The dramatic effect of the story (and the value of the system to its users) was heightened by the information that the person in charge of the project had advocated increasing the company's investment in Web technology and that he was under pressure to prove that his recommendation was a good one. A later scene involved one of the company's public relations representatives, who gets a call from a journalist asking her to comment by five o'clock that evening on possible health effects of the technology they were planning to use to monitor the athletes' locations. The public relations person had not heard of the project before, but knows she will be blamed if a negative story appears in the press. With sweat running down her spine, she uses the system to locate the people working on the triathlon project to determine if any health issues exist and to prepare a response for the journalist.

The Benefits of Using Stories in the Design and Development Process

Stories are valuable artifacts throughout the product design and development cycle for both cognitive and social reasons. From a cognitive perspective, stories represent events and experiences in a coherent way through schemas that capture the relations and structures connecting individual details (Schank, 1990). This facilitates comprehension and memory and enables the recall of similar events with similar underlying structure (Cohen, 1989; Klatzky, 1980). For example, a story of how a complaint with a toy was handled by one company might remind someone of how a problem with a room was resolved by a hotel, despite the fact that there is little overlap in the specific details. From a social perspective, stories are a key mechanism through which human experience has been shared for generations. Given the right social environment (such as a group of colleagues over lunch recounting encounters with problem customers), people can find sharing stories to be a natural, effortless, and compelling experience (Ochs & Capps, 2001). Stories are thus powerful tools not only for capturing the situations in which technologies will be used, but also for encouraging others to recall relevant situations from their own experience.

A focus point for customers. Because of their cognitive and social power, stories can be valuable tools for customer research and user feedback. Stories focus customers on their experience instead of on an unfamiliar technology, prompting more accurate, detailed, and honest feedback. A common reaction when someone is shown a story set in one domain is to say, "The same thing happens here, only it's different." This provides an opportunity to probe deeper, have the person tell the story of his or her similar experience, and discover places where a depicted technology might need to be modified. Stories can help uncover potential problems with a planned design that stem from personal or cultural differences; as people hear a story they can reflect on whether it feels natural and if they could see them-

selves doing the same thing. Field studies and ethnographic research familiarize designers with the specifics of the people and settings for which they design, much as an author would engage in background research before writing about an unfamiliar domain.

Due diligence. Because stories involve many of the elements that will be important for a successful product, the act of creating a story can be a way of ensuring that critical issues have been addressed. Constructing a story at the early stage of a project can ensure that the research homework necessary to design an effective solution has been done. Furthermore, it helps the team understand the results of such research, not just as individual pieces of data, but in a coherent, causally connected way. To phrase it somewhat harshly, if you cannot tell a compelling story about how a system you are designing will be used and the value it will bring to the people who will use it, you should question why the system is being built in the first place. In practice, we have found that much of the effort required to construct a story is effort that the product team needs to make anyway. Constructing a story can ensure that this work is done early in the process.

Prompts for innovation. Stories can be used to prompt innovation as designers, forced to consider individual users with specific goals, characteristics, and situations, uncover opportunities to enhance the user's experience in novel ways. Thinking about a specific person performing an activity in detail and asking, "How can my tool help the most at each point" can point to features and functionalities that were not part of the initial conception of the product. We have even found this approach to lead to innovation on the level of business model, when, for example, the team asks a question like, "Wouldn't our protagonist prefer to just get that function as an externally hosted service?"

Establishing a shared vision. Stories are also effective tools for teaming and establishing shared vision around a project and for addressing many of the problems associated with multidisciplinary software design. Stories are understandable across the different cultures and languages of software business. Because they are not the traditional tools of any one of the established software development disciplines, stories are readily accessible to all members of a multidisciplinary team. Using stories in multidisciplinary storyboard sessions helps teams overcome politics and cultural differences to create a shared vision around the customer.

Story-based methods carry the customer goals as a reference thread throughout the product lifecycle. Used at all decision points, they can ensure a focus on providing customer value where technology would otherwise be the deciding factor (what's easier, more available, etc). They can also serve as a lens through which to see and explain business problems that otherwise would be overwhelming, and can be an effective tool for conveying the value of a design to marketing.

Compelling stories, compelling designs. Designing a story to show how a new offering will be used captures and represents these elements in a more cohesive, comprehensive, and meaningful way than do lists of functional requirements or proposed features. To some extent, a story can be seen as a template to ensure that the team has done due diligence on thinking through each of the elements required for a successful project and then combined them into a whole with the proper relations described.

Used in this way, stories and storyboarding serve not just as a tool for communication, but as a methodology for design. In essence, designing the story is really a way of designing the offering itself. With large, poorly constrained, multifaceted problems (multiple market segments, many possible product and service offerings, multiple business models, many dimensions of demographic and market data), it can be especially useful to focus in detail on designing for specific, fleshed-out, individual users and settings.

Management buy-in. A critical success factor for software development projects is that not only do the developers feel confident about the product (which is mostly based on technical details), but also that all levels of the management support it. Stories can help tremendously with that because their focus is not on technical details, but on a level that is easy for everybody to understand. All persons involved in the project, including development, marketing, planning, and management, can understand and agree on the common goal of the software development project that the story describes.

Starting point for other product externals. Stories are very useful in all phases of a software development project, from the initial research, task analysis, and audience definition to design validation. They can, however, be used even more—as a starting point for product externals such as tutorials and marketing flyers.

For example, after we used a story early in the development cycle to validate the task analysis and audience definition, we immediately recognized a lot of interest in this approach, especially from marketing and information development. Marketing often needs a way to explain complex software products in an easy-to-understand way to potential customers. In this case, the story was an excellent starting point for them to create flyers and marketing material. The same is true for online documentation and tutorials, where there is a need for easy-to-understand examples to explain the product. Therefore, a story can be used not only to support the software development process, but also as a perfect starting point for product externals.

Two Major Types of Stories

This article describes two approaches to creating stories and suggests when and how to use each. It is the reader's decision which approach would be best to accomplish his or her goals.

Fictional stories. Fictional stories are created by the design team based on the design team's best understanding of the goals for a future vision solution. Fictional stories can be useful when the design problem is a new space, for instance, when you are creating a solution for a problem that customers do not yet know they have. For instance, imagine trying to describe the usefulness of a television or a FAX machine to people before they ever anticipated the need for such a thing or had a concept of what those devices were. It would have taken a story to help customers see beyond their daily activities to the use and value of these inventions.

We have found fictional stories to be effective tools for eliciting feedback from potential users of a technology, even at its early stages. Fictional stories can benefit teams in the very early stages of vision and strategy phases, especially in areas of new technology and newly combined cross-divisional solutions. Because of the flexibility and immediacy of this approach, fictional stories offer a very attractive option for simply initiating the story-based approach and getting a team to think in story form. If used at the market definition phase where business cases are explored, the process of creative storymaking can uncover new business uses or potential audiences for a particular solution.

Although it is best to involve storymakers who have some understanding of the intended audience, a team that is somewhat naïve in this regard can offer completely fresh ideas. The recommended follow up when using this approach, however, to validate the stories and related storyboards with real customers before investing time and resources in further development.

Customer stories. Customer stories are based on customer experiences. One approach is to use the real stories and anecdotes you have gathered, as is. Another method is to visit multiple customers and extrapolate across common subjects to create a *metastory*.

Using a single story intact can work when you want to illustrate a specific case or when business processes are well established and you know that there is little difference in the practices between customers. The extrapolated approach is effective when you need a more encompassing description than exists at one customer site. It is also useful when there is a need to protect the privacy of customers and a directly recorded anecdote is too revealing. Finally, it is a good approach to use for generalizing the roles or tasks to orient customers away from individual details and validate a high-level process.

When the customer is intimately familiar with the tasks and the associated problems they encounter, it is best to take this data and use it to construct real customer stories. Customers have a mental model of their work, however inefficient that might be today, and it is best to respect this and use it as a place to begin, even if you change aspects of it in the end. Given their current tasks and processes, they will have lots of anecdotes about what they are currently doing, what they wish they could do, and what happens when things go wrong.

Later, when doing solution validation with them, the customer stories approach is also valuable. When customers recognize that they have a problem and can critique a proposed solution based on that experience, they can react concretely and

specifically to this reference point and use it to tell us whether the solution will solve their problem or not.

Customer stories may require more planning than fictional stories; there must be the time and resources to plan and document real customer stories. However, we have found cases where the real customer stories are more dramatic and compelling than anything we could have imagined.

3. EXAMPLES OF THE USE OF STORIES

This section presents examples that illustrate the value of using stories at the various stages of product design and development. Ideally, stories can be used throughout all phases of a project. However, not all projects are ideal, and teams frequently need to choose the phases in which they can introduce the use of stories. For this reason, we have chosen specific examples of how we have used stories at various phases of the design process. In some cases we also include actual stories that we have used. The final example shows how stories were used throughout the entire design project and how the story evolved in each of the design phases.

Although we assume that the general phases and sequence of the design process we describe here are familiar, it is useful to point out that people take personalized approaches to it. You will see some evidence of our personal approaches in the following examples. And, because these examples come from the teams who created them, they are each personal accounts by nature.

3.1. Stories in the Design Process

For the purposes of this article, we define the four main phases of the design process as *research, design articulation, evaluation, and execution*. (See Figure 1 for a complete illustration of this process.)

Because stories come in many flavors, it is important to analyze the story content used at the various stages to determine if a particular story has the correct characteristics for the phase of which it is a part.

Stories in the research phase describe the problem as defined by the research into customers, technology, and business parameters. The stories resulting from the research phase that feed into design include who the customers are for this solution, what the technological requirements and timeframe limitations are, what is happening in leading solutions today, and so forth. We have called the stories used in the research phase *problem statement stories*.

For the design phase the team focuses on the solution, so aspects of their future vision will be apparent in a story used in this phase. We call the story in the design phase a *solution story*. The characteristics of a solution story are the same as a problem statement story, but with the pros and cons of the technical environment and the details of the flow and composition of a proposed solution added. A storyboard may become an added story artifact in this phase. A storyboard is a series of visual diagrams to illustrate what a user might see and do, accompanied by a narrative “script” of the

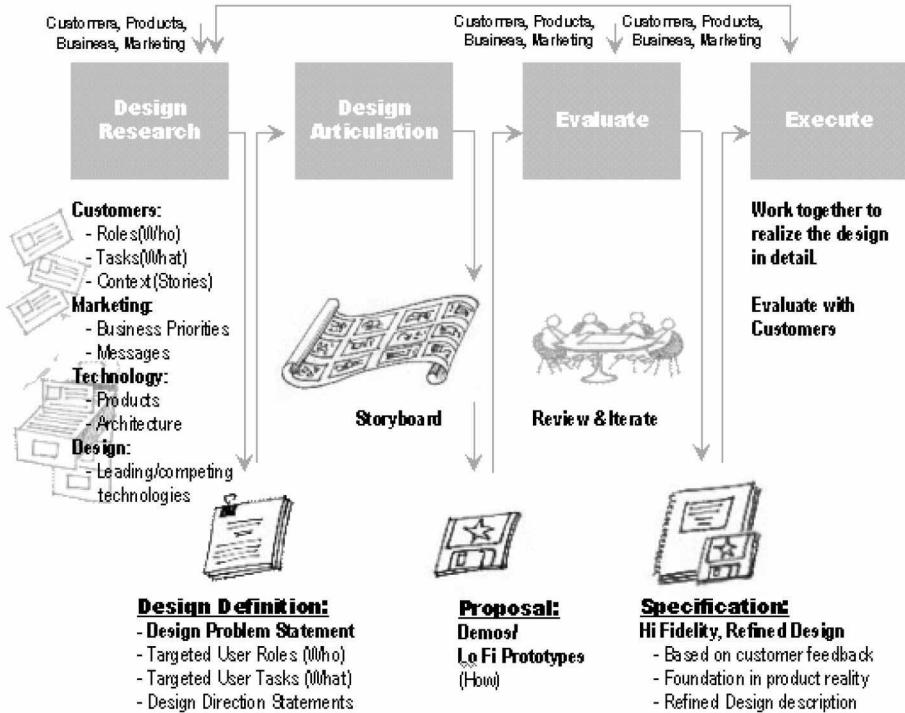


FIGURE 1 The design process.

story. Because of the benefits of exploring multiple solutions, the team may produce several storyboards or document multiple paths in a single storyboard.

During the evaluation phase, customers, product team leaders, marketing representatives, and key managers discuss revisions to the solution and make agreements and qualifications. These documented decisions bring the unique content to the story here. These rounds, which happen more than once and often include different groups, uncover new technical and task questions and problems to be solved. To articulate these changes, the team may choose to create more detailed descriptions and prototypes. They may also produce detailed visual design and information architectures. In the end, through this series of dialogs, what may begin as a solution story changes into what we call a *detailed product design flow*.

In the execution phase, stories and storyboards are used as a blueprint. For this reason, a story at this phase will have a high level of technical and customer detail. There will also be a level of agreement to the solution across the organization. Although it is still possible to encounter technological problems that would require fundamental changes to the solution, most of the core problems will have been covered by this point. With the technical assessment and details fleshed out, the story now becomes a *product design specification*. This can be thought of as a detailed prototype and accompanying scenario that focus on the steps the user would need to

take to accomplish a particular goal. In addition to the technical details, the visual design and information architectures will have their own detailed designs as part of the product design specification. So although the story is still the framework, by this phase it is almost outweighed by all of the accompanying material produced to support it.

3.2. Research Phase: Aiding Customer Research

A common problem in gathering customer research is dealing with customers who want to jump right into the solution discussion. In this case, stories can focus them on the details that define the problem. For example, you can ask the customer to describe the last time they had a problem with “x” or something similar. This will usually produce a compelling story. Another strategy is to ask customers to imagine things by saying, “Let’s pretend that resources and technology are no object. Now, if you could have anything you wanted to be able to do ‘x,’ what would it be, and what would it be like?” In response, they will tell you a sort of story. If they begin to describe what you have been designing, you can sketch something for them and ask if this is what they had in mind.

3.3. Example 1: Conveying Findings From Customer Interviews

The following is an example of a metastory that was assembled from several customer experiences. The story was created, based on these real situations, by the customer research architect to represent all the elements of the software design problem at hand.

Bill is a software distributor working for the IT [information technology] System group in a large global bank. Generally speaking, his job is to distribute software packages across functions at the bank. This can mean sending new versions of software out to the banks’ employees (i.e., a new version of Office) or patches to existing versions of software when they are published by the software companies. Usually, these kinds of distributions can take place during the day with no disruption to banking operations. However, in this case, the software being distributed is a large core-business application and needs to be distributed after normal banking hours, when the check processing system can be shut down for 4 hr.

Everyone expects this to be a routine distribution. However, at 2 a.m. on Monday morning, toward the end of the distribution, Bill finds himself staring at a flashing error message that is outside the scope of his knowledge and experience. He clicks on the help button, but it is no help. He goes to the online help for the distribution tool, and discovers that they do not publish the error codes online. He realizes that it might not be a problem with that software, but with some of the network software. Because Bill is not a network specialist; he is really in over his head.

Bill knows that if Jack were here, this would be no problem because Jack is the team’s source of obscure bits of technical trivia. Jack always knows the right person to call. Bill rushes to Jack’s cubicle to rummage through Jack’s sticky notes, stacks of tech-

nical documentation, company procedures, contact numbers, and shelves of bound manuals. He calls some of the contact numbers, but no one answers. One of the contact numbers is no longer valid. He finally calls security to have them unlock Jack's desk to see if there are any more up-to-date clues on what do to, which wastes an additional 45 min. After 2 hr of searching, Bill gives up. Nothing there seems obviously applicable. He faces the cold fact that in 3 hr his management will be there, and there is no time left to complete the distribution. Bill's only hope is to get the old version of the software running in time for the check processing system to be up by the start of banking hours. If only the information he needed had been placed where Bill could find it easily.

Research phase: Task and audience definition. Task analysis provides core information on which to base an offering design. It focuses on understanding users, the environment in which they work, the tasks they currently perform, the tools they are using to do their jobs, and how they imagine they would like to do their work in the future.

Because stories are told from the perspective of specific individuals, they can ensure that a team considers the elements critical to the success of a system. Teams working from traditional design documents such as functional specs might otherwise ignore these elements. Following are two examples of projects in which creating story elements, including detailed characters and settings, problems to solve, and motivated actions led the team to see their audience and tasks in new ways.

3.4. Example 2: Defining Audience and Tasks for a New Product

Large organizations often purchase software through paid subscriptions that allow a specified number of people to use the software for a set period of time. Subscriptions are renewed periodically (typically annually), a process that often includes modifying the mix of products, number of users, and other provisions of the contract.

The group responsible for subscriptions for a major software product wanted to create a Web front end to allow customers to purchase and renew subscriptions, and to let customers manage their accounts online. They also wanted to open the subscription model to smaller businesses. Early on, the group decided to create a storyboard showing the various screens the customer would access to perform basic tasks such as selecting, verifying, and modifying details of an existing account; understanding alternatives; obtaining pricing information; and renewing subscriptions. They believed the storyboard could help justify funding for their planned Web design.

The group had an extensive flowchart (Figure 2) posted along the wall of their meeting area outlining the process the company followed internally to initiate and renew software subscriptions. They planned to use this process as the starting point for the structure of the Web site by adding a Web screen or dialog box for each task in the flowchart.



FIGURE 2 The flowchart covering an entire wall, depicting the internal process required to manage software subscriptions.

When we discussed creating a visual storyboard with this group, we emphasized that a critical element was getting the story right. This meant knowing in detail who would use the system, why they were interested in the software in the first place, what their concerns and goals were, and so on. When we first asked the group who their customer would be, they replied, “a small to medium-sized business,” citing the rather generic terms that appeared in their marketing studies. The customer’s goals were similarly described broadly, in terms such as “reducing cost” or “speeding access to information.”

As we worked with the group to define the detailed setting, characters, goals, and problems needed for a compelling story, questions arose that pointed to the need for the group to include participants from other departments (including sales and customer service) and to conduct a series of customer interviews.

Ultimately, we created a story around the fictional setting of a growing playground equipment manufacturer, which was representative of the intended users (Figure 3). The problems faced by the company included the cost of updating and maintaining printed catalogs and materials. The cast of characters in the story ranged from the company’s nontechnical CEO, to the accounting person who handled purchase orders, to a sales manager whose salespeople were complaining about the difficulty of getting up-to-date sales materials to their customers (Figure 4). Each of these characters was given a name, a face (through photographs), and a set of goals and concerns.



FIGURE 3 A depiction of the company created as the setting for the story around the new software subscription system. Though fictional, details were representative of actual companies and intended users.

A major contribution of the story was the realization that the team needed to think beyond a monolithic customer to consider the individual needs of the various people at a customer site who would be involved in purchasing and managing a software subscription.



FIGURE 4 The characters in the software subscription story.

3.5. Example 3: Modeling Users and Tasks for a Follow-Up Product

The goal of this project was to deliver a follow-up version of MQSeries Workflow, a software tool that supports the modeling and execution of business processes and related IT and organizational infrastructure activities. This face-lifted product would reflect a changed marketing focus, address new market segments, and accommodate new usage scenarios.

It was important to know from the beginning of the new development cycle whether our existing task analysis and audience definition would still be valid for the changing market needs. The MQSeries Workflow product is very complex. It has many functions and different user interfaces for installation, configuration, administration, modeling, and monitoring. This was acceptable for the existing version of the product because different users in a company used different parts of the product. However, would this also be true for the new target audience? And, would the new users expect these different functions to be integrated into their existing software frameworks such as the administration console of Microsoft's Management Console presentation service?

Faced with these concerns, we needed to do more to define the audience and tasks than simply list them in a table. Instead, we wanted to verify our assumptions, not only with our existing customers, but also with users who did not have much experience with workflow software. To explain the complex product to the various target audience groups, we chose a story-based approach.

We presented the story using an interview style. In the story, we asked people (representing the new tool's audience) in a fictitious company what they wanted to do with the product (their tasks), what their current problems were, and how IBM MQSeries Workflow could help them with their job in the future.

Because we wanted the product vision to serve sales and support personnel as well as the primary users of the product, we also looked at the design of the product externals, such as marketing brochures, product presentations and Web pages—things that are often neglected when developing a new product. As we designed and wrote the story, we tried to incorporate as many of these product externals as possible. To enhance the story, we used existing screen captures or prepared mockups of the newly designed user interface to gather early feedback. The story was written in HTML so that it could be distributed and viewed readily, making it easy for people to evaluate the product concept. Figure 5 shows one of the pages of the story.

We asked existing and new customers as well as the sales, competence center, and support teams within IBM for feedback, both on the product and the overall story approach. The feedback we received was very positive and covered all aspects of the story; for example, ease of use of the defined tasks as well as the audience description and the relevance of details in the story.

Here is a sample of feedback from two of our customers:

Your media is a very good way of communicating and connecting to the content very easily. Very good opportunity to pin down the content and design of the products to users' expectations and tasks.

The screenshot shows a web-based application titled "Workflow Vision". At the top, there's a navigation bar with links: Introduction, Overview, Use Cases, Product Support (which is highlighted in blue), Summary, Consumer, Workflow Administrator, Customer Care Agent, Service Engineer, Call Center, Business Analyst, Software Engineer, and System Administrator. To the right of the navigation is a small graphic of a person holding a briefcase. Below the navigation, there's a section titled "Software Engineer" featuring a portrait of a woman named Kathrin. A question is posed: "Q: Kathrin, could you tell us about your responsibilities as a software engineer?" An answer follows: "A: Our team is responsible for developing and integrating the applications we need to run our business processes, which are defined in our workflow system. We also have to ensure that the data routed between applications is available in the correct format, which means that it must be transformed before it can be used by an application. For this, our system setup also includes MQSeries Integrator together with MQSeries Workflow." Another question is asked: "Q: You seem to be committed to achieving ambitious objectives. Can you give us a few details about some of the tasks involved?" An answer follows: "A: I'm involved in several projects, for which I have these responsibilities: • Creating and customizing our in-house applications • Adapting our applications to run smoothly in the workflow environment". On the left side of the main content area, there are two sections: "Deliverables" (Development environment, Modeling Tool, Online help, APIs, API Reference Manual, Programming Guide) and "Functions" (Application integration, API search function, Developers forum).

FIGURE 5 Sample page of the story.

I very much like the fact that the vision is driven from the user's expected experience. It describes what the vision is now from what the user experiences (performing their business tasks) are expected to be when the product is available.

The story helped our multidisciplinary development team gain new insights on their new audience and tasks beyond the feedback they had received on the existing product design. The story provided high-level descriptions, which helped us at an early stage to get a common understanding of what the product had to offer. This helped us check our assumptions, both with customers and within the development, marketing, and management teams. At the end of the process, the teams felt reassured and were glad to have a common vision for the new version of the product.

Design and development phases. Starting with design and continuing through development, practitioners from a variety of disciplines need to collaborate closely. This is where the value of a story in providing a shared vision and common language truly comes to light.

Different disciplines have different ways to approach a project based on their training, experience, and specific goals for the design process. To use three common players as examples, developers, marketers and human-centric designers often see the world through related lenses, but with vastly different emphases. Developers are the ones who will be tasked with actually building the software, and

so they traditionally have a very pragmatic, schedule-based approach with functions and requirements as their baseline measurements. Marketing professionals focus on customer segmentation, pains, and messages. Human-centric design professionals exist somewhere in between, concerned with tasks and users. Good professionals in these areas will be passionate about the importance of their view of the world, and sometimes this passion can lead to conflict.

Stories provide a neutral avenue for representatives from each of these disciplines to articulate the key aspects of the concerns they see, and to weave them together into a shared vision. For instance, a solution story will have its context based on the audience targeted through marketing research, and will be based on the key pain points that the marketing arm determines. These pains will be detailed on a human level in the story through user and task descriptions. And, technical details from development will fill in these problem spaces described in the story with achievable solutions. Using the story based approach, all three disciplines can drive the design from their perspective. All three can satisfy their need to control that the design is on the right track from their perspective.

How does that happen? Stories help to “put a face” on the user for all the varied team members. Stories weave together laundry lists of requirements into a cohesive whole that reveals the customers’ motivations and priorities. These motivations and priorities help designers make decisions as they discuss design trade-offs with practitioners from other disciplines. They help software engineers understand for whom they are developing and why. The concepts of personas and stories go hand-in-hand. Defining a persona (Cooper, 1999), or type of user, provides each member of the team with someone concrete to understand and identify with and helps him or her understand more about these types of users (e.g., their work environment, training, and preferences). Attaching stories to that persona makes them real. It becomes much easier to ask, “How would this design solve Sue’s problem?” when put into the concrete context of a story.

Sometimes there may be several different audiences with specific tasks that must be addressed. In these cases, stories can treat each audience and task set as a distinct group, allowing designers and developers to focus on designing to meet each group’s needs.

3.6. Example 4: Giving Functional Requirements Context and Priority for Developers

Working on EDMSuite, a tool for Enterprise Document Management, we visited a customer and followed several of their users around, watching them work. The customer had a number of content management products that dealt with online documents of various types, mostly forms and images. Our job was to find ways to integrate the products into a suite that would make it easier for users to perform their tasks than the individual products did. We discovered some wonderful integration opportunities in gathering our stories. One particular user, Sue, was working with an online document in one of the applications. At one point, Sue printed the document and walked over to the printer. She then picked up the printed docu-

ment, walked over to the scanner, and scanned it into another product. When we asked why she had just done this, especially when both applications worked with the online documents, Sue replied that that was the only way she knew to get the document from one of our applications to the other.

Sue's story clearly illustrated to our team the importance of providing integration between the products in the solution. If the solution to the problem had been simply listed as just another requirement, the development team's motivation for implementing this feature might have been diluted. When we can all see why our solutions are necessary in the real world and how the requirement ties back to our customer's business, the urge to solve the problem becomes compelling.

Stories are also important in the design phase because they put things in user terms rather than technological or business terms, or in terms of any particular implementation or solution. Maintaining the user perspective in the early stages of the project provides the space and mindset to explore various solutions. In this way, stories can open the team to creative possibilities. Too often, design teams jump straight from an idea to its implementation without fully exploring whether there are alternative solutions or if the idea addresses the root problem to solve. Without guidance to the contrary, we naturally try to define things in terms that are familiar. Customer stories can help provide the guidance to go beyond initial conceptions.

3.7. Example 5: Stories as a Bridge Between Marketing and Development

This example illustrates the value of the story-based design approach in establishing strong communication links to marketing. Communicating product direction from one discipline to another is not an easy task. Everyone is trying to do the right thing, yet the differences in language, approach, and intent between marketing and development can create stressful friction at times. Even more dangerous is the fact that we sometimes found ourselves building functions to solve the wrong customer problem. This is where we have found stories and storyboarding to be unexpectedly useful.

At Tivoli, the marketing organization creates a market requirements document (MRD), which outlines the product's market, business problems, and goals. Ideally, the MRD should outline the parameters of the problem, including the constraints, at a level of detail that will leave the design possibilities open for the design process. However, it is a challenge for the authors of the MRD to maintain an objective view of the problem space when they receive input from vocal executives and customers about what the solution should contain. Technical details often creep into the MRD and become product directive, even though the underlying problem might be solved better in another way.

This example describes how the Tivoli Presentation Console design team introduced a skeptical marketing team to the idea of story-based design and as a result gained powerful advocates for the approach. Initially, the relation was tightly managed by the marketing executives who were very concerned about our interaction with customers and wanted to have a great deal of control over every conversation. As is appropriate, we were directed by marketing to focus first on the change management problems as the highest level of market opportunity. Our marketing team

helped us select the customers to visit, scope the problem space, and guide the main points of the research.

Because the marketing team was understandably cautious about including new people in their customer relationships, we needed to gain their confidence in our ability to work with their customer contacts. We were careful to include marketing in each phase of the design. For example, we formally presented and gained their approval for our research approach with customers, got their buy-in on the resulting stories, and invited them to participate in the storyboarding. Later, when they saw that the type of research we were doing had nothing to do with selling solutions and was focused on the problems their customers were having, they trusted us to manage the customer communication and resulting documentation without their constant presence.

We maintained this close communication with the marketing team while we framed the design goals. When marketing saw the resulting storyboard, it was clear that it illustrated the customer problem in a way that they understood and approved. Moreover, it illustrated a high-level direction for a solution that they could easily understand. Whereas the MRDs of the past contained laundry lists of technical requirements, this approach, which was created in partnership, provided a vision of the design as a whole. Marketing was ecstatic at how the storyboard helped them communicate a cohesive vision clearly to customers, as well as to others in the company. And, because this storyboard was created in collaboration with the development leads, it was technically feasible, accurate, and apolitical.

The storyboard then became the MRD. Gone was the thick text document, replaced by an illustrated flow created by the multidisciplinary team. Although it included technological information that made it real, understandable and applicable to the development audience, it did not constrain the development teams to an unnecessary level of detail. It showed a general design direction while the actual detailed design was left to be solved by the experts. Our marketing representatives went so far as to require that all future Tivoli MRDs be in the form of a storyboard. They had become not only converts, but advocates.

Using stories through all phases of the design process. In the introduction to this section, we defined the types of stories that can be used in each phase of the product process. The following, extensive example gives the reader a view into a possible approach to an overall story-based design process. It also illustrates how one story is transformed from a problem statement story to a product design specification by the different needs and considerations in each phase of the process. The reader can see here the details and artifacts resulting from each phase that make the transformation process valuable, while maintaining connection to the original customer accounts.

3.8. Example 6: A Story Evolves Throughout Design

This example illustrates the evolution of one of a series of stories through the design process for the Tivoli Presentation Framework Console. Because our customers

needed to use the console with many different products, we had to look at how information, security, and decisions were affected from product to product and from user to user. As a result, our stories spanned several products and involved many users who interacted. The integration layer between products is an area with a high risk of design failure and, therefore, one where stories can be their most powerful.

Our story comes from research on the entire flow of how large enterprises plan and execute the deployment of new software into existing business systems. In doing this research, we observed the exchange among eight types of users of Tivoli software: the executive administrator, coordinator, application expert, inventory expert, package builder, distributor, Level 3 help, and end user. Although our final story spanned eight pages and all eight of the aforementioned roles, we limit our example here to pieces of the distributor, change coordinator, and Level 3 tasks. An ellipsis (...) indicates portions that have been omitted.

Our original problem statement story is the result of exploring the problem during the research phase. It begins this way:

Banking U.S.A.'s executives have mandated that everyone in the corporation will have PeopleSoft on their machines by June. This company has a well-defined process for scheduling changes, so

The distributors, George, Jake, and Ian, go through the process of doing the detailed schedule of the software distribution for next weekend after the affected employees have gone home. They determine which machines need which software packages. They now have to go check to make sure that the targeted machines are available. There are many details in this task, such as how much bandwidth they have available, decisions on what to do about single failures, and doing a contingency plan.

Next weekend, Ian, the distributor on duty, gets notified that the distribution is due to start automatically in 1 hr. As the distribution is happening, there is a team, including the change coordinator, Bob, who will check the progress of the distribution. Ideally, Bob would like to be able to monitor the distribution remotely and intervene during the process if necessary.

As it turns out, one of the repeaters fails during the first hour. Ian sees the problem on his display, opens a problem ticket, and calls the Level 3 on duty. Stan, the Level 3, has already detected a problem with a server from his monitors. This server, which contains the repeater in question, has crashed. Stan reboots the server, checks to see if the problem is resolved, and closes the problem ticket

The story is transformed into a solution story through the design articulation phase, when our multidisciplinary team of human factors, visual design, development, information architecture and marketing representatives described the solution they envisioned. It was key that all or most of their perspectives were incorporated in real time at this stage because each had unique creative input, and their exchange enriched each other's ideas and the total solution. In our example, we began with the customer story and added the solution proposals at a high level to produce a storyboard. The skeleton of the storyboard script was the original customer story; the flesh came in the form of additional technical details.

The solution story that follows is actually our second try, and this gives us the opportunity to make an important point about this approach. It is vital to have a

solid mechanism for getting early customer feedback on design direction. Our design team members had done a lot of early thinking about the change management space, and the design direction contained a lot of well-founded preconceptions. Once we had a storyboard and script for our original proposed solution, we took it to customers to validate. Because the design direction was so clearly articulated in the storyboard, it was easy for the customers to understand this solution in terms of their environment and business processes. The customers did not agree with our initial proposed design direction and gave us specific feedback about where it fell short. In fact, they proposed a fairly radical redirection of our efforts. Because the design was simply a storyboard at this point, it was easy for the team to change direction. The importance of this lesson cannot be overstated. Even with a very experienced, knowledgeable, and creative cross-disciplinary team, we were reminded that we are not the experts; our customers are. No matter how good a team is, sometimes design instincts are wrong, and story-based design helps detect that early and economically. Instead of wasting valuable time and resources producing a product that would not have met customer needs, we were able to change course quickly, develop another storyboard and script to illustrate a different proposed solution, validate it, and move forward—all before any code was written.

Our second storyboard and accompanying narrative were as follows:

.... after Sally, the executive administrator, creates the change ticket

The distributor, George, gets a beep and opens his to-do item. Here, he has not only the software packages and the endpoints, but also the calendar reference and announcement note generated by the change coordinator, Bob. When George opens the distribution scheduler, he sees the master schedule. He should be able to drag and drop the packages onto the schedule. He uses Bob's calendar as a reference to know where to target. Henry, the package builder, has already created the packages to know what their bandwidth requirements and other constraints are. Therefore, George can overlap the desktop client installs so they run simultaneously with the server installs but independent of them. The Web client "knows" which must be done before others can be completed. Further parameters and constraints are set, as well as notification preferences. George can drag the announcement out onto the calendar as well so that it will be sent out 1 or 2 days ahead of the distribution. All this will happen 3 weeks from that point (see Figure 6).

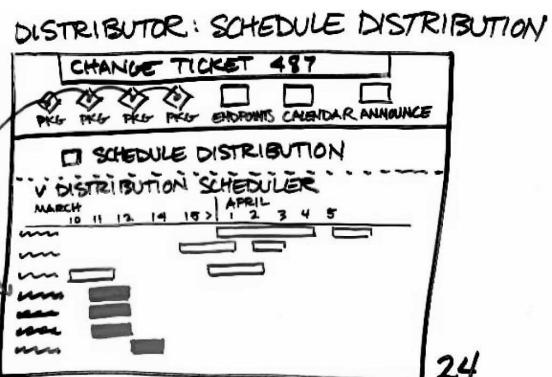


FIGURE 6 Distributor's view—schedule distribution.

Three weeks later, Bob is monitoring some distributions at home from a Web browser. From here he has a go-no go capability to stop the distribution for any reason. He can monitor the progress of the distributions. He also has quick access to the distributor's beeper number and can chat with the distribution team. The chat is contextual with whichever distribution is selected. Different distribution teams may work on different items, and he would want to chat with more than one potentially at one time (see Figure 7).

Meanwhile, back at the office, Ian, the distributor on duty, is doing some job scheduling when his to-do list beeped. He opens it up and sees that the Peoplesoft distribution is scheduled to take place that day. He also sees (Figure 8) that he has several other distributions that he needs to monitor

Bob is still monitoring from home. He may not have the fancy topology view available from the office, but at the very least he should be able to see (Figure 9) the same information about progress indicated in a table.

At this point, the distribution runs into a snag. One of the repeaters stops working. The software distribution product becomes aware of this event and makes Ian aware through the topology view. He also gets important error details on his Tivoli Assistant card. It tells what happened as well as the impact and gives him the option of opening a problem ticket. Ian creates a problem ticket and sends it on. Although we do not see it, we note that the problem ticket should be automatically populated with contextual information such as the name of the trouble repeater and the submitter name, based on the fact that the ticket is launched from this error message (Figure 10).

COORDINATOR: MONITOR DISTRIBUTION

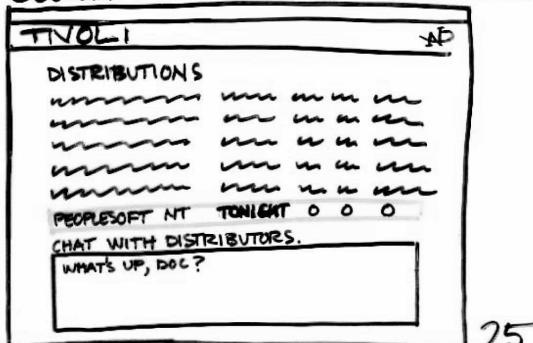


FIGURE 7 Coordinator's view—Monitor all distributions.

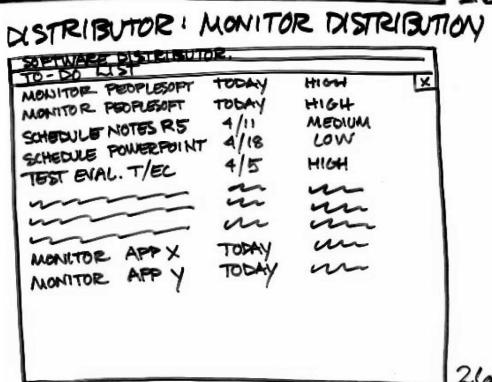


FIGURE 8 Distributor's view—Monitor all distributions.

Stan, Ian's favorite Level 3 guy, receives the problem ticket and opens it. On the problem ticket itself, he has an attachment to the troublesome repeater, which has a red status (Figure 11). Directly from there, Stan should be able to do appropriate corrective action for a downed repeater, such as remotely rebooting the machine it sits on. After the reboot, the repeater is green, and Stan gets a message that it is back up and running. He has the option here to close the ticket or go on and do more diagnostics using a wizard, depending on which answer he gives

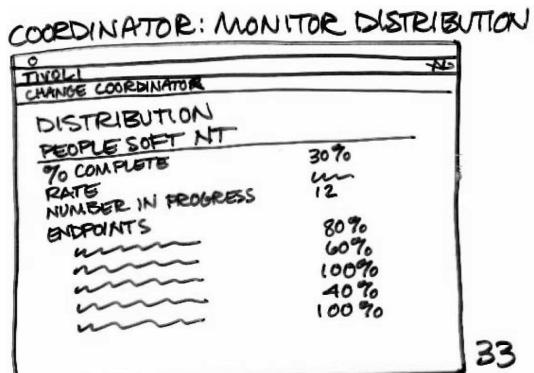


FIGURE 9 Coordinator's view—Monitor single distribution.

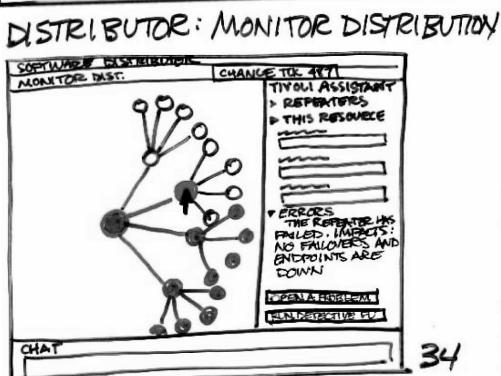


FIGURE 10 Distributor's view—Monitor single distribution.

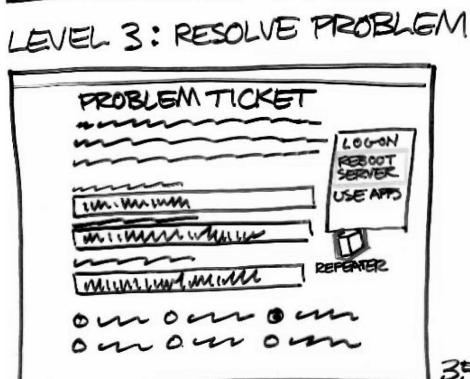


FIGURE 11 Level 3's view—Resolve problem.

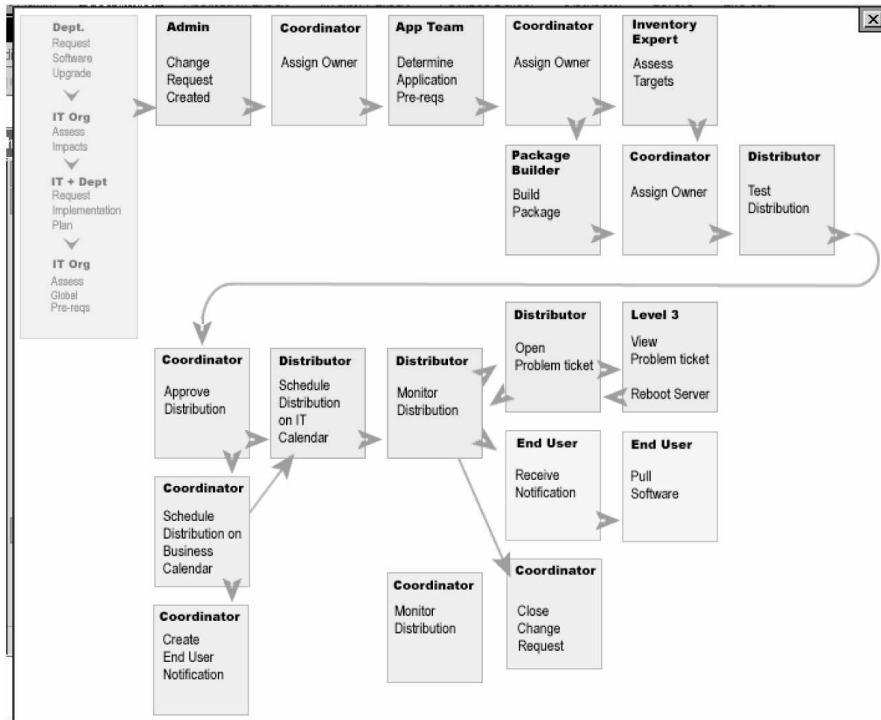


FIGURE 12 The process flow.

In the previous example, the lack of detail in both the words and the pictures is intentional. That detail will come later in the process. Note also that we picked a task flow that we thought would have broad applicability to our customers, as a major goal was to present a proposed solution to gather customer feedback.

Next, in the evaluation phase, the team mapped out the process flow from person to person, as this was considered to be critical to the whole solution (see Figure 12). Through evaluation with executives, customers, and extended team members, the solution story changed into a detailed product design flow. Although the form of the story did not change significantly during this phase, it underwent significant revisions. At the same time, we were also producing a version of the visuals that would include more details on the content represented, some initial interaction cues and flows, and initial style concepts. Where the original storyboard might have only a box with squiggly lines to indicate a control, in this phase the designers determined what kinds of objects and controls to use, what the relations between them should be, and initial directions for visual design and information architecture.

We rendered the storyboard in more detail and added interactivity using Macromedia Director, using essentially the same narrative. Many of the details we added were pulled from real customer examples and things they said they wanted to see. Later, you can see an illustration of some of these pieces of the detailed product design flow.

...George, the distributor, gets a beep and opens his to-do list. Here, he has not only the software packages and the endpoints, but also the calendar reference and announcement note generated by Bob, the change coordinator. When he opens the distribution scheduler, he sees the master schedule. George can drag and drop the packages onto the schedule. He uses Bob's calendar as a reference to know where to target. Henry, the package builder, has already created the packages to know what their bandwidth requirements and other constraints are. Therefore, George can overlap the desktop client installs to run simultaneously with the server installs, but independent of them. However, the Web client "knows" that it cannot happen until the others are finished. Further parameters and constraints as well as notification preferences are set. George can then drag the announcement out onto the calendar as well, so that it will be sent out 1 or 2 days ahead of the distribution. All this will happen 3 weeks hence (Figure 13).

Three weeks later, Bob is monitoring some distributions at home from a Web browser. From here, he has a go-no go capability to stop the distribution for any unexpected reason. He can monitor the progress of the distributions, as well as have quick access to the on-duty distributor's beeper number, and can chat with the distribution team. The chat is contextual with whichever distribution is selected. Different distribution teams may work on different items, and he might want to chat with more than one team at a time (Figure 14).

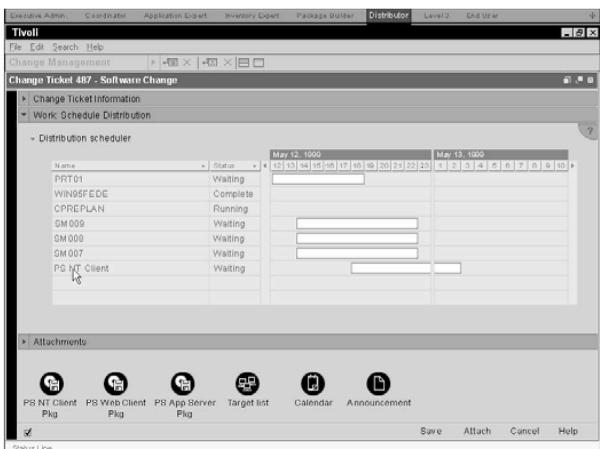


FIGURE 13 Distributor's view—schedule distribution.

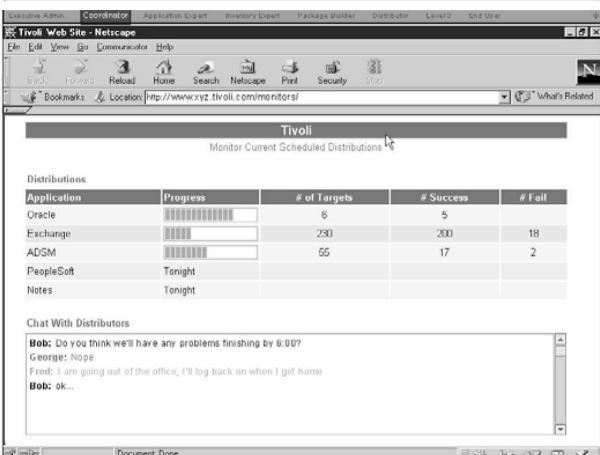


FIGURE 14 Coordinator's view—Monitor scheduled distributions.

Meanwhile, back at the office, Ian, the distributor on duty, was doing some job scheduling when his to-do list beeped. He opened it up, and sees that the Peoplesoft distribution is to happen today (Figure 15). He also sees that he has several other distributions that he needs to monitor

The Peoplesoft distribution begins, and Ian is monitoring it (Figure 16)

Bob is still monitoring from home. He may not have the fancy topology view available from the office, but at the very least he should be able to see the same information about progress indicated in a table (Figure 17).

At this point, the distribution runs into a snag. One of the repeaters stops working. The software distribution product becomes aware of this event and makes Ian aware through the topology view. He also gets important error details on his Tivoli Assistant card. Here, it tells what happened as well as the impacts and gives him the option of opening a problem ticket, which he does. Ian creates a problem ticket and sends it on. The problem ticket was automatically populated with contextual information such as the name of the trouble repeater and the submitter name, based on the fact that the ticket is launched from this error message (Figure 18).

Stan, his favorite Level 3 guy, receives the problem ticket and opens it. On the problem ticket itself, he has an attachment to the troublesome repeater, which knows enough to have red status. Directly from there, Stan should be able to do appropriate

This screenshot shows the Tivoli Distributor interface. The main window title is 'Manage Distributions - Monitor Active Distributions'. It displays a table titled 'Red Team Distribution Progress' with the following data:

Application	Progress	Scheduled Start	Priority	# Targets
Oracle	[Progress Bar]	10:00 AM	High	20
Exchange	[Progress Bar]	1:00 PM	Medium	120
ADSM	[Progress Bar]	4:00 PM	Medium	15
PeopleSoft	[Progress Bar]	6:00 PM	High	

Below the table, there's a note section with a timestamp 'Tonight' and a status 'Medium'. A 'Chat with Distributors' pane shows a conversation between 'PeopleSoft Coordinator' and 'Jake'.

FIGURE 15 Distributor's view—Monitor active distributions.

This screenshot shows the Tivoli Distributor interface with a topology view for 'Monitor PeopleSoft 487'. On the left, a network diagram shows various computer icons connected by lines. On the right, a 'Tivoli Assistant' window is open, displaying the message: 'One or more repeaters are not reachable'. It includes a progress bar and statistics: 'Total number of targets: 16', 'Targets completed successfully: 6', and 'Targets completed with error: 0'. A 'Search For:' input field is at the bottom.

FIGURE 16 Distributor's view—Monitor PeopleSoft distribution.

corrective action for a downed repeater, such as remotely rebooting the machine it sits on. After the reboot, the repeater is green, and Stan gets a message that it's back up and running. He has the option here to close the ticket or go on and do more diagnostics using a wizard, depending on which answer he gives (Figure 19)

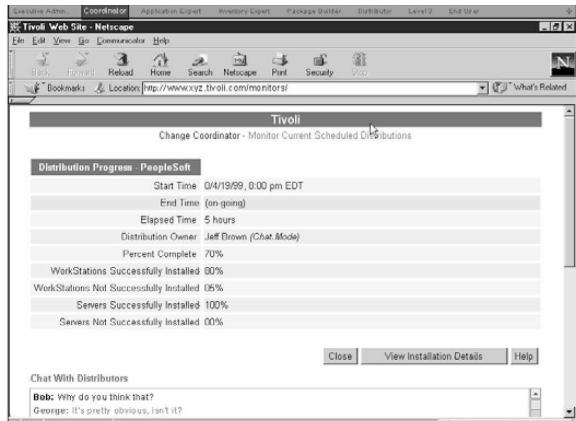


FIGURE 17 Coordinator's view—Monitor PeopleSoft distribution.

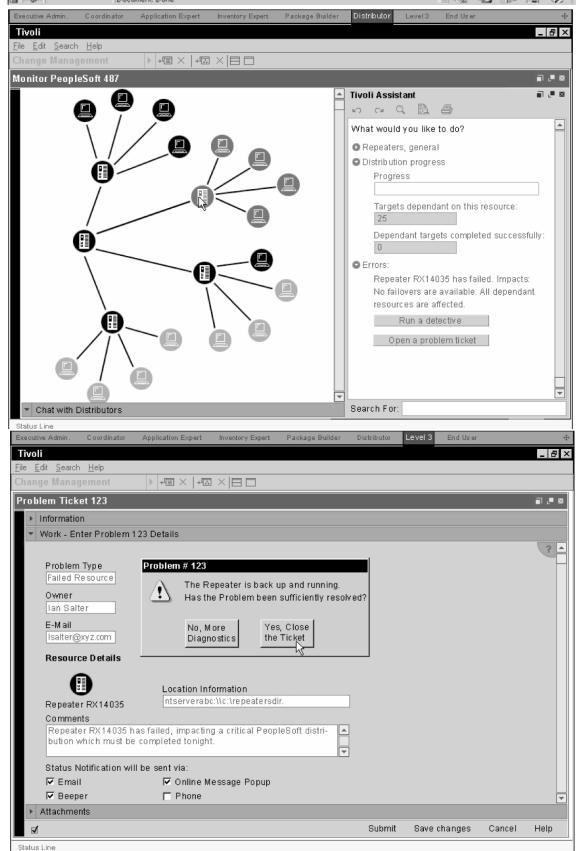


FIGURE 18 Distributor's view—distribution problem.

FIGURE 19 Level 3's view—problem ticket.

For the final incarnation of the story in the execution phase, namely the product design specification, the physical artifacts depend on how the individual product development team works.

In our case, the development team adopted the storyboard and reverse-engineered it to determine what the technical requirements would be and who would work on it. Initially, they worked directly from the hand-sketched storyboard illustrations and added their own notes to the panels. (Unfortunately, no documents showing this use of the storyboard still exist.) As the design progressed, we produced many more artifacts to explore details of the design. These examples illustrate the level of discussion that occurred between the designers and developers during this phase.

Figure 20 is a section of the design specification that documents details of interaction. These details were all based on information from the original customer research and stories.

Universal Navigation and Windows Management

Users can alternate between these views through some interface element, be it the “view” item on the menu bar or a toggle switch on the toolbar of the console. All of these views should be available, and the console should remember which view each user most recently used.

The MDI “workspace” model: (Figure 20).

In which there are several subordinate windows floating in a master navigation workspace or window.

In addition, there was a need to decompose and label the overall design so it could serve as a roadmap for the cross-divisional teams of developers who were learning the design and would need to code to it. Figure 21 shows one example of the design in progress, with labels showing what the individual elements were and how they would be used.

Such details as how the status line should be rendered are covered at this stage. For the developer, every small design element has many design questions that, ideally, they will want to discuss with the entire design team. Figures 22 and 23 show what existed already and the proposed redesign. Many detailed design questions

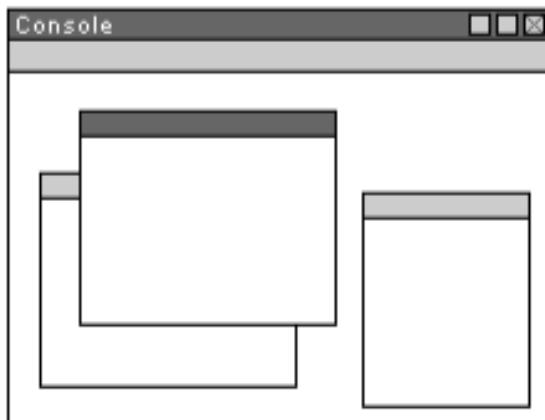


FIGURE 20 Illustration of design specification.

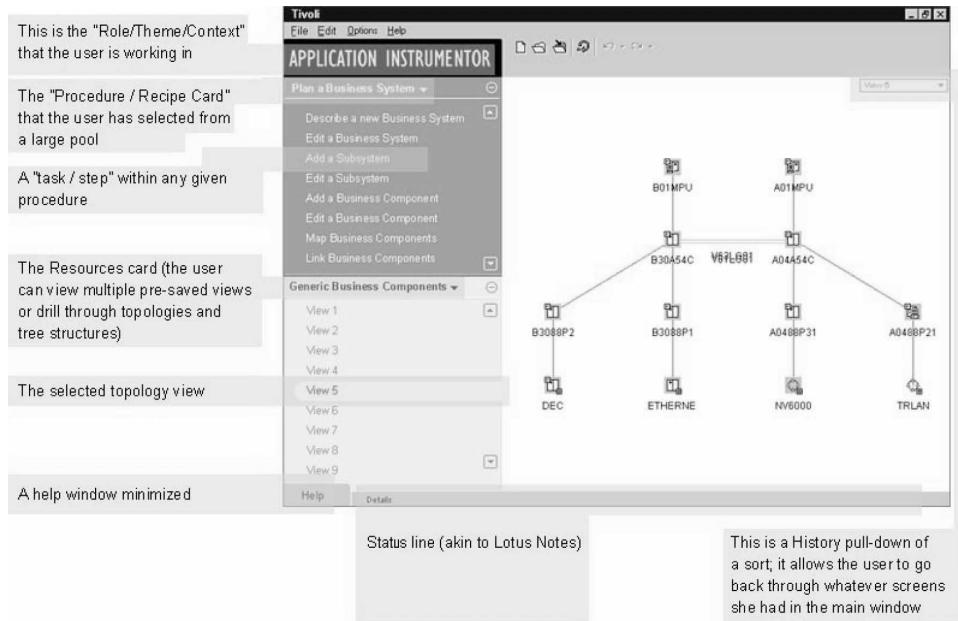


FIGURE 21 Design in progress.



FIGURE 22 Status line, before.



FIGURE 23 Status line, after.

arose in this phase, such as, What information should be presented? Should the information be in formal or informal voice? Should there be icons and how should they look? Our stories helped here as well.

4. CONSIDERATIONS WHEN USING STORIES IN DESIGN

Three considerations frequently arise when using stories as design tools. The following recommendations are based on our own experience and conversations

with teams who were considering using stories in their design work but expressed common reservations.

4.1. Accuracy

As discussed previously, a story used to help guide development and design must accurately reflect situations and concerns in the real-world settings in which the technology will be used. Even if not true in every detail, the essential elements that would impact the design should be accurate. A common fear is that a team will be led astray designing to a story based on a naïve or incorrect view of what really happens in the user community. To protect against this, it is critical that stories be informed by significant real-world observation. In the case of fictional stories, this means involving informants who are knowledgeable about the task domain and setting in crafting the story. Equally important is the need to validate the story by reviewing it with people in the task domain and modifying it to reflect their experiences.

4.2. Generalizability

A related concern is the extent to which stories are generalizable and reflect the breadth of situations in which a technology will be used. Because stories are, by definition, about specific users and settings, there is a fear that a story could lead a team to focus on a single setting and activity at the expense of others. On the other hand, there is a conflicting fear that without a story, it would be hard for the team to really understand how the technology they are building will be used. There are several ways of addressing these concerns:

1. Use the results of field and marketing studies, demographic analyses, and other sources of data to understand the range of domains and situations for which the technology is expected to be used. Use that knowledge to guide the selection of settings and activities for the stories you create.
2. Do not rely on a single story. Instead, design several stories to ensure that you cover the key settings, domains, and tasks in which the technology will be used. Working with a series of different stories can point out the extent to which a single tool can satisfy all the broad needs and the extent to which the tool must be flexible and customizable.
3. Understand that stories are not the only tools to use in the design process. They should be accompanied by field studies, usability testing, surveys, and the like.

4.3. The Ease of Using Stories Effectively

Although most people can understand, appreciate, and empathize with a good story, not everyone knows how to write one. Having people with experience using stories on the team is valuable. Fortunately, we have found that people can learn to

be better storytellers and storycrafters through practice and training, and by becoming sensitive to the elements that lead to effective stories. People who have used stories in the past can help the team understand the right level of detail for characters, settings, and plot lines. They can help ensure that plot lines effectively probe the value of the tools being designed and the broader motivations and goals that they are meant to accomplish. They can also help determine the most effective way of representing the story (as text, visual screens, or a movie) based on audience, goals, and constraints on time and resources.

5. CONCLUSIONS

Stories are powerful tools for ensuring the overall value and positive user experience of designed systems. In many ways, designing a compelling story means doing much of the same work one must do to design a compelling solution. For stories to be effective, they must include many of the same narrative elements that contribute to a compelling novel, short story, or movie. In practice, this means that teams using stories as a design tool can benefit from including participants with experience or skill in crafting stories.

As we have shown through our examples, teams at IBM have found stories useful at every stage of the development process, from early vision and innovation, through customer research, audience and feature definition, prototyping, user interface design and development, and marketing and rollout. This strength is based on the value of stories as rich tools for capturing and communicating human experience combined with the social power of stories that makes them an effective resource for teaming, eliciting feedback, and sharing common vision. The story-based design process not only knits the solution to the real problem, but can weave the team's contributions together in a way that builds the level of communication and trust of the team. When each discipline can see their contribution to the end solution, the level of satisfaction and empowerment grows.

Because effective storycrafting requires a sensitivity to human issues that are critical to overall user experience, thinking in terms of stories has made us better, more perceptive designers and developers. We suggest that the bookshelf of every user experience designer contain, along with texts on HCI, cognitive science, ethnography, graphical technique, and a few good books on crafting stories.

REFERENCES

- Brannigan, E. (1992). *Narrative comprehension and film*. London: Routledge.
- Burroway, J. (1999). *Writing fiction: A guide to narrative craft*. Reading, MA: Addison-Wesley.
- Carroll, J. M. (Ed.). (1995). *Scenario-based design*. New York: Wiley.
- Cohen, G. (1989). *Memory in the real world*. Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.
- Cooper, A. (1999). *The inmates are running the asylum*. Indianapolis, IN: SAMS.
- Klatzky, R. L. (1980). *Human memory: Structures and processes* (2nd ed.). San Francisco: Freeman.
- McKee, R. (1997). *Story: Substance, structure, style, and the principles of screenwriting*. New York: HarperCollins.

- Ochs, E., & Capps, L. (2001). *Living narrative: Creating lives in everyday storytelling*. Cambridge, MA: Harvard University Press.
- Rosson, M. B. (1999). Integrating development of task and object models. *Communications of the Association for Computing Machinery*, 42, 49–56.
- Schank, R. (1990). *Tell me a story: Narrative and intelligence*. Evanston, IL: Northwestern University Press.
- Schank, R. (1997). *Virtual learning*. New York: McGraw-Hill.

Copyright of International Journal of Human-Computer Interaction is the property of Lawrence Erlbaum Associates and its content may not be copied or emailed to multiple sites or posted to a listserv without the copyright holder's express written permission. However, users may print, download, or email articles for individual use.

A UX Maturity Model: Effective Introduction of UX into Organizations

Lorraine Chapman and Scott Plewes

Macadamian Technologies

Gatineau, Quebec, Canada

{lorraine, scott}@macadamian.com

Abstract. Getting products out the door with a fantastic user experience (UX) is becoming increasingly more important in all aspects of the business world. Large companies have raised the bar in consumer products in terms of UX design, which has leaked into non-consumer organizations and contexts. The same people, who are also consumers, are now going to work with equally high expectations in their enterprise applications or even using their “consumer” product at work. Naturally, organizations that create products have responded by hiring consultants or professional UX designers. Yet, despite having the right skills, organizations are not necessarily getting the results they want. Achieving great UX design is not just a function or talent of *individuals*, it is an organizational characteristic. Understanding the organization’s “maturity” level is a necessary first step for improving the effective delivery of UX design and for enabling the organization to advance to the proverbial “next level.”

Keywords: UX maturity, maturity models usability, user-centered design, user satisfaction.

1 Introduction

“I think in order to design great products, you need to have the culture in place.”

Cordell-Ratzlaff, 2010

Organizations are more and more seeing the value of hiring user experience (UX) professionals and incorporating user-centered design. More and more programs at universities are cropping up than ever before, to meet the demands in industry. Large name companies such as Google and Apple have incorporated UX design as a centerpiece of their successes. There have been a number of well-known treatises on the introduction of usability into an organization. For example Shaffer’s work [1] documents some of the key components organizations must have in place to institutionalize usability. While not in conflict with Shaffer’s ideas, here we focus more on how to effectively recognize what stage an organization is in with regards to usability and the key areas of focus for transitioning to the next stage.

There is a belief in the business value of UX design that simply did not exist 15 – 20 years ago [2]. There are more individuals trained and capable of excellence in UX design. However, beliefs and skilled UX resources are not enough to ensure that the UX design aspirations and goals for an organization are met. The user experience of a product is not the result of a UX designer alone – it is a result of how the organization as a whole executes on the product creation. There are a number of ways of decomposing the elements of user experience design and success. For the purposes of this paper, we will consider the following construct: Product value and user experience largely comes from two essential values.

1. An *emotional* value ("I love my new smartphone.")
2. A *utilitarian* value ("I can call and text my friends from anywhere.")

These two values are linked for most products in the sense that attaining a positive emotional response – over the long term – is difficult without at least some utilitarian value. We see emotional and utilitarian values delivered by three aspects of design:

1. Aesthetics

The product literally “looks good.” People will often describe the aesthetics in non-scientific terms such as “slick,” “clean,” and “cool.” When explored more deeply, these characteristics indicate at least some elements of aesthetics influencing their responses [3].

2. Functionality

The product allows the user to accomplish a goal that they value. For example, a user can scan their computer for a virus, make a phone call, or print a document. However, having functionality that is of value by no means ensures that it is easy to accomplish, easy to discover, more efficient or satisfying to use.

3. Usability

Usability is the manner in which functionality is delivered and experienced by the user. Is it effective, efficient, satisfying, and simple? [4]

It is not uncommon for some companies to focus primarily on the utilitarian value of the product and, in particular, on the functional aspect of the product design. For many product development program, success is defined by the number features that can be fit into each release cycle. Yet a great product delivers a user experience that combines aesthetics, functionality and usability to meet both the user’s emotional and utilitarian needs. In discussing UX maturity, we are really postulating the following two key points:

- Organizations higher up on the UX maturity scale are more able to deliver effectively to the combination of functionality, aesthetics, and usability that is appropriate for their business goals.
- Organizations at certain levels of maturity implement processes, capabilities directed towards improved success in UX design that is appropriate for their level of maturity. That is a “stage 2” company, cannot jump to a “stage 5” design process.

There is an evolutionary process associated to moving through the different stages of UX maturity.

Knowing and understanding an organization's true UX goals and its level of maturity can guide decisions ranging from where an organization seeks help (by hiring internally or using consultants), to what processes they use and implement, and how decisions are made.

In our experience, there are six key indicators of UX maturity.

1. The *timing of UX* involvement in the design and development process. The earlier UX is involved, the more mature the company.
2. The *UX expertise and resources* in house and/or ability to bring in UX expertise quickly as needed.
3. The use of appropriate *techniques and deliverables* to obtain and understand user input and capture UX design.
4. The *leadership and culture* in the company. How well the leaders and company as a whole appreciate the value and necessity of UX design from a business perspective.
5. The degree to which UX processes are *connected and integrated with other* corporate processes that enable individuals to work together to create the user experience of the product(s).
6. *Design thinking is applied in the broadest perspective* possible to drive consistent customer experience.

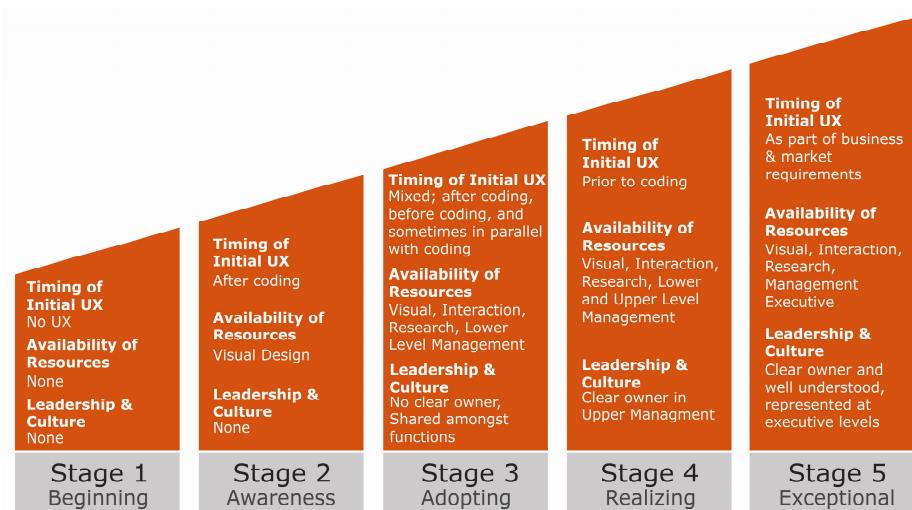


Fig. 1. The Stages of UX Maturity Model

Considering these criteria, we have created a model for assessing an organizations level of UX maturity (see Fig. 1). This is one model by which organizations can assess their level of UX maturity; however, there are other ways of looking at the UX maturity of a company, other factors or weighting of factors that could be considered [5]. The key factor is using models that can help with understanding a complex situation, making decisions and gaining insights [6]. The relevance in this model is less about exactly what stage an organization is in, and more about the insights and ideas decision-makers and influencers can utilize by applying this model to their own organization. For this reason, rather than focusing on absolutely measuring maturity, the model focuses on indicators of maturity and relating these indicators to stages of UX maturity. The intention is that the indicators will spur decision makers into action. If an organization aspires to “stage 5” UX goals, but there are several indicators that suggest the organization is at “stage 2,” decision-makers can use the indicators to assess how they may progress to the next level.

2 Stage 1: Beginning

To the extent an organization thinks about UX design at all, at this stage an organization will typically see it simply as visual design. Design is perceived as something to be applied on top of the product’s functionality. For software, it is generally addressed at or near the end of coding. There are no professional UX designers in house; often developers are responsible for the UX design. There may not even be product management, or product management is relegated to making suggestions about the design or functionality to developers. Essentially this means all the key indicators are “dark”; there is no expertise, techniques, or design-thinking culture that would suggest UX design is even being attempted in a meaningful way.

2.1 Implications

Products must differentiate themselves primarily on functionality or other factors outside of the direct user experience. This includes customer support, salesmanship, or the ability to technically integrate with other products or systems. Products created in this way are at high risk of displacement if competitors are able to match their primary value propositions while differentiating through a superior user experience. For example, an information management system at a hospital may be valued and purchased because it can “hook into” other hospital management systems. However, if it is cumbersome to use and causes significant overhead to the personnel who are interacting with it, another compatible product offering a better experience of information management could eventually displace it.

2.2 Key Signs of an Organization at This Stage

- UX design is almost never talked about or it is discussed only in terms of graphical design.
- Users are not consulted in the product design and development process.
- If any user suggestions or complaints are gathered, they are not critically evaluated. They are either dismissed or implemented verbatim.
- Any UX design activities have very little formal structure (developers do the necessary elements of screen layout) and are probably not considered part of a UX design activity.
- There are no UX design goals tied to business objectives.
- Product management may be non-existent or minimal.

2.3 Critical Success Factor to Achieve the Next Stage

Taking the first step towards UX design involves ensuring that the relevant business issues are correctly identified as being UX design-related. This awareness is often created through a combination of a significant “shock” event (for example a competitor wins a major sale because of their UX design) along with some degree of education and awareness occurring within the organization.

3 Stage 2: Awareness

At this stage, the organization may be considering UX design, but applying very little structure around UX activities. Often there is a significant amount of misunderstanding surrounding the real nature of UX design. Therefore, usually there are no UX professionals employed in the organization, though an outside "guru" may be brought in or consulted. UX design improvements or insights gained during the product design phase are often implemented in bits and pieces in the final product.

3.1 Implications

Out in the marketplace, the products must still differentiate themselves primarily on functionality or other aspects of the business beyond the direct product experience. The risk of competitive displacement remains high if competitors can match the key value propositions while differentiating themselves on user experience. Organizations at this stage must decide to what extent they wish to invest in introducing formal UX processes and practices in order to forestall the competition.

3.2 Key Signs of an Organization at This Stage

- UX design is a "hot" topic of debate for at least some projects.
- People are starting to make design decisions or suggestions based on articles, a conference or seminar attended or a personal interpretation of UX.

- Employees conduct design reviews with considerable discussion, yet become frustrated with the limited progress being made and disagreement on how to actually resolve UX design issues.
- Most user requirements are confined to marketing input or functional improvements.
- There is little user feedback or it is limited to asking users their opinions on design or functionality.
- UX design goals are general or hard to measure (for example, "the interface should be intuitive and straightforward").
- A UX professional is consulted during a project too late in the process.
- There is inconsistent awareness and buy-in to making UX design investments, such as training, beyond a few people.

3.3 Critical Success Factor to Achieve the Next Stage

In this stage, organizations have an awareness of UX design having potential business value, but minimal to little understanding of what that means. Education is required to pivot the organization to higher maturity. While training is useful, it typically does not include decision makers and, therefore, an organizational understanding of UX design is not achieved. At this stage, organizations should seek to launch a pilot project, overseen by experts, with a clear connection between UX design goals and a business objective. It is this connection that pulls in the key decision makers and allows them to see the value of UX design; otherwise, organizational maturity may stagnate.

In our experience the transition from this stage to the next is the most difficult and most crucial, largely because it requires the most significant change in company. As the Director of Product for a company in the aircraft industry put it, "this is simply not in our DNA". Not only was it necessary to determine the right activities and projects to focus on, but also at least as much time has been (and is being) spent understanding individual challenges, agendas, concerns and perspectives of key decision makers and stakeholders. The effort is less on understanding what the right logistics are, and more about incorporating and dealing with change in roles and perspectives.

4 Stage 3: Adopting

An organization at this stage is experiencing the growing pains of adopting more sophisticated UX practices; some projects run smoothly and experience successful outcomes, while others fail miserably. Typically, there is growing belief among the leadership team of the value of design (although the leaders themselves may have little knowledge of UX or mixed knowledge of UX) and investments are being made in professional hires or contractors/firms. The prospects for using UX design as a competitive differentiator are positive; however, there is a high risk of getting stuck at this stage, or worse, "regressing" to old product design habits. This happens if the occasional project does not go well or new leadership comes in and acts to revert progress in maturity. This stage definitely has expertise in house or consultants involved,

although it may not always be making the best use of the resources or UX resources are significantly overloaded in terms of deliverables and responsibilities.

4.1 Implications

Some products are now distinguishing themselves based on UX design, or at least they are not losing to competitive alternatives because of UX shortcomings. Success is still inconsistent across the brand (as is the design), and customers may not associate the company itself with excellence in UX design. However, the organization is starting to realize tangible difference to the business because of the benefits of an improved UX to their products.

4.2 Key Signs of an Organization at This Stage

- There have been some successful products recently where the UX design has clearly had positive business impacts.
- UX goals are measurable and clear on some projects. For example, the organization may be establishing user error rate targets, efficiency measurements, and user satisfaction goals.
- Users are regularly consulted on many projects; although perhaps not always in the right ways or in time to inform design decisions.
- The organization still experiences some issues with UX projects. It may be that good UX design is created on paper, but not fully realized in the product.
- There is no senior leadership or management in UX. The UX function may report into Marketing, Product Management or Engineering or is distributed between individuals across projects.
- There is no standard design and development process being practiced across the organization. For example, some projects or parts of the organization may insist on usability testing of products, while others do not.
- Similarly, roles are not standardized throughout the organization. On one project a business analyst may be responsible for understanding user context and motivations; on another project it is a UX researcher.
- There is a lot of discussion about UX design within the organization. Successful products/projects that involved UX design, and competitors who have used UX design as a differentiator, have made executive decision makers take notice. These executives and influence leaders now have strong and differing opinions on product design.
- A common perspective on UX design does not exist throughout the organization. The expertise exists within the company at the project delivery level, not at the executive level.

4.3 Critical Success Factor to Achieve the Next Stage

When it comes to organizations at this stage, moving to the next level means setting clear UX goals for teams on products/projects and requiring accountability from UX experts, as well as assigning empowerment to UX experts. However, the UX experts only are one key component. Related roles need to be defined so that everyone on a project/product feels they can contribute to UX outcomes on a project. Obviously senior leadership and experience is required to help align and co-ordinate UX resources and other functions. Among the best next steps is to augment the management and executive team with senior UX leadership and understanding. If this is not done, then while a desire for quality UX results may be in place, the lack of understanding at the decision-making level can undermine the experts at the less senior levels.

Moving through this stage through to stage 4 and even beyond is less of a challenge of people and change management and more about refining and adding nuances. As a medical company we worked with – who already had a well defined UX process – pointed out, our real value came in enhancing aspects of requirements gathering, subtleties in research methodologies and interpretation of data. This is significantly less challenging than a fundamental change to process and roles. It was more about additional skill sets and methodologies. They then could then incorporate and further on their own for future work.

5 Stage 4: Realizing

Stage 4 organizations, those displaying excellence in their UX design maturity, have moved far beyond discussing and arguing about whether or not it should be part of design and development. They are now more concerned with the nuances or particulars of improvement in UX. UX goals are clearly embedded with the organization's mindset and people understand their roles in the process.

5.1 Implications

The company has a reputation for UX excellence in their products and often wins on this point. It is used as a selling proposition and a differentiator. The company brand is clearly linked to great UX design.

5.2 Key Signs of an Organization at This Stage

- Many examples of successful products where the UX design has clearly had positive business impacts.
- UX goals are measurable and clear on almost all projects.
- UX is no longer viewed as hot topic, rather it has become table stakes. Discussion around UX is more likely to be about the latest techniques, process improvement and how to better incorporate it at all levels of the organization. Discussions amongst key stakeholders are no longer about where UX fits in a process or who is responsible for making design decisions.

- There is senior management leadership and accountability for UX at the same level as product management, development, marketing, sales, and other functions.
- A strong set of practices, processes and guidelines exist that are consistently utilized by project teams.
- If excellence in UX design is sacrificed, it is an intentional trade-off driven by business goals with well-understood consequences.
- Users are regularly consulted for product/projects. Design research with target end users is done consistently with correct techniques.
- UX design is considered at a "product family" or portfolio level and the decision-making processes and development are organized in recognition of this need. Each product is designed with other products that the customer might also use in mind so that the transition is seamless because consistent UX designs are used.
- All functional areas (such as Product Management, UX, Marketing, Engineering) are certain about their roles in the UX design process and understand each other's roles. While UX design experts make UX design decisions, ideas and innovation come from the entire team involved in product design and development.

5.3 Critical Success Factor to Achieve the Next Stage

Organizations at this level have a highly effective level of UX design and it is difficult to move beyond this level of maturity. We do believe, however, there is a natural evolution a company can make. Whether or not it is strictly an evolution of UX design is debatable, but we include it here for completeness. The next step beyond excellence is when the organization realizes that the product experience is just one part of a larger experience delivered to customers. Organizations must start planning complete customer experiences that include, but go beyond, the UX of the product alone. However, to move beyond stage four means understanding from a customer point of view what the experience is with the company as a whole and how the product fits in with that experience. This includes all of the touch points and processes that happen around the product such as, discovering it, buying it, installing it, using it, upgrading it and “sun setting” it.

6 Stage 5: Exceptional

When an organization is exceptional in its strategic implementation of UX design principles, UX design is firmly integrated into all aspects of customer experience – these organizations have fully realized their UX maturity goals and design thinking truly permeates all aspects of the organization. The same type of thinking that went into designing the product experience is present across the board in all customer touch points, although different experts and roles, not just UX designers, will implement it.

6.1 Implications

The company has a “gold standard” reputation for excellence in all aspects of customer experience (for example, marketing, sales, support, product design). This reputation is significant in achieving business goals and maintaining a competitive advantage.

6.2 Key Signs of an Organization at This Stage

- All aspects of Stage 4 product-orient UX excellence are strongly entrenched.
- When designing a product, the whole "ecosystem" of a user's experience with the company/brand is considered.
- UX goals are linked to business objectives with the total customer experience in mind.
- The first thought on UX design in a project is probably more about the overall customer experience rather than the intended user experience for just the product. This manifests itself in up front UX and customer experience research and the creation of artifacts such as customer experience maps.
- There are senior leaders accountable for customer experience and this part of the organization works with all functional groups that create/deliver customer touch points such as, UX designers, marketing, sales, and support.
- Research in UX is strongly coordinated with other customer experience feedback processes.

7 Conclusion

As discussed in the introduction, this 5-stage framework may not exactly match the specifics of a particular organization. In the real world, organizations may display a mix of characteristics from different stages across their organization. Nevertheless, we believe assessing a company against some of these key indicators can provide insights into opportunities and issues that will allow a company to adjust its trajectory and attain its business aspirations that are dependent or related to successful user experiences practices and their execution. Organizations, like people, learn by doing. And to do properly experts need to be hired or consulted. Identifying projects, deliverables, activities are key to progressing. Training, attending talks, reading books, is certainly helpful, but not enough. UX design is no different than any other function in this regard.

References

1. Shaffer, E.: *The Institutionalization of Usability: A Step-By-Step Guide*. Addison Wesley, New York (2004)
2. Trenner, L., Bawa, J. (eds.): *The Politics of Usability: A Practical Guide to Designing Usable Systems in Industry*. Springer, Berlin (1998)

3. Lavie, T., Tractinsky, N.: Assessing Dimensions of Perceived Visual Aesthetics of Web Sites. *International Journal Human-Computer Studies* 60, 269–298 (2004)
4. International Organization for Standardization: Ergonomic requirements for office work with visual display terminals (VDTs) – Part 11: Guidance on usability (ISO Reference No. 9241-11:1998(E)) (1998)
5. Vetrov, Y.: Applied UX Strategy, Part 1: Maturity Models. UX Matters (2013),
[http://www.uxmatters.com/mt/archives/2013/12/
applied-ux-strategy-part-1-maturity-models.php](http://www.uxmatters.com/mt/archives/2013/12/applied-ux-strategy-part-1-maturity-models.php)
6. HIMSS Usability TaskForce.: Promoting Usability in Health Organizations: Initial Steps and Progress Toward a Healthcare Usability Maturity Model. Health Information and Management Systems Society (2011)