A study on user-centered design illustrated by development of social platform for dance classes attendees

Tomasz Legutko

2016/2017

Contents

1 Motivation

Software development methodologies have always been and always will be an important part of all IT projects. With every year software becomes increasingly

important part of our society and with every year the software's complexity becomes even harder to manage. Over the years there's been a great progress in architectural patterns, tools, best practices and particularly, development methodologies and frameworks.

Famous waterfall. Agile Manifesto. XP and Scrum. No silver bullet, especially with usability.

Usability, HCI.

Startups as particular area of research. [18]

Usability and Agile is well researched and well documented.

Merging of those domains has little literature - hence the reason for case study. It's important because will provide guidelines for industry practitioners.

2 Introduction

2.1 Brief history of software development process

Nowadays, agile methods are without a doubt an IT industry standard. But many think of them as just a replacement for waterfall, while iterative and incremental development dates as back as to mid-1950s, with few well documented undertakings by IBM and NASA [14].

2.1.1 Waterfall

Waterfall became well-known after 1970 Winston Royce's article "Managing the Development of Large Software Systems". It promoted well-defined, strict process, consisting of gathering requirements, analysis, design, coding, testing and maintaining. Even though original article advised following those activities twice, waterfall was most known as it's single-pass version and became a standard for long years, especially in large-scale and enterprise projects. Craig Larman [14] argues that, as famous H.L.Mecken quote says "For every complex problem, there is a solution that is simple, neat and wrong.", waterfall gave illusory sense of "orderly, accountable, and measurable process, with simple, document-driven milestones", was easy to explain and recall (much easier than iterative and incremental development practices already present at that time) and widely promoted in literature.

2.1.2 Iterative and incremental development methodologies

Back in 1970s iterative and incremental development (IID) approach was already becoming recognized as more natural and fitting to manage projects than waterfall, notably IID incorporation by IBM Trident submarine system with above 1 million lines of code and NASA's space shuttle software. In 1976 "Software Metrics" Tom Glib proposed "evolutionary project management", and argued that system should be implemented in small steps, producing the appearance of stability in order to "have opportunity of receiving some feedback from the real world before throwing in all resources intended for a system".

1980s consisted of vast amount of criticism in literature towards unquestioned dominance of waterfall in industry [14], advocating IID approach, with notable 1985 Barry Boehm's publication "A Spiral Model of Software Development and Enhancement". It introduced risk assessment and project review in each iteration along with development phase using appropriate management model. During that time there were well documented significant project failures by US Department of Defense using document-driven waterfall approach, which resulted in adjusting formal standards (DoD-Std-2167A) to encourage usage of IDD.

In 1990s, software development industry was much more aware of IID practices and there were many books, articles and standards promoting IID. In second half of 1990s most currently known methodologies were created - Scrum, eXtreme Programming (XP), Lean Software Development, Rational Unified Process, Dynamic Systems Development Method and Feature Driven Development.

In 2001, a group of 17 methodologies experts, including all of above mentioned met in Utah to discuss common practices and the term "Agile" was coined.

2.1.3 Reigns of the Agile

Meeting in 2001 resulted in famous Agile Manifesto [2]:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

Agile clearly promotes lightweight process and has been well received in industry and increasingly adopted, with recent 11th Annual State of Agile Report in 2017 [15] stating that over 94% of over 20,000 respondents claimed their organizations practiced agile. Among agile practices, Scrum is undeniably the most commonly chosen methodology (58% respondents) with Scrum/XP hybrid taking the second place (10%).

Since the creation of agile, the industry has also moved forward by leaps and bounds in technical areas. Automation is now common in software process - quality assurance, continuous integration and continuous delivery, internet as development environment, distribution means and execution infrastructure. Not to mention great progress in languages, tools, frameworks and architectural solutions. [10] The progress is unquestionable and although large IT projects were once notorious for spectacular failures (famous E.Yourdon's "Death March" [28]), reported project success rates (meeting budget and time constraints and project goals) in various studies from went from barely 20-30% in 2000s [13] to 60-70% in 2017 [19].

2.2 User Centered Design

User Centered Design is part of Human-Computer Interaction domain, which primary focus is on designing and creating interactive systems with the aim to make them user-oriented and usable. It's a very broad domain, focused not only on computer science and engineering, but incorporating knowledge from various research fields such as cognitive and social psychology, human factors and ergonomics, industrial design and many others. Historically it's been occupied with trying to grasp the nature of human interaction with machines. [22]

In 1950s the approach was called "System Ergonomics", it offered a holistic approach - systems and users were seen as single interacting system. In 1960s and 1970s there were "Cognitive Systems Engineering" and "Socio-Technical Systems Design". Cognitive Systems Engineering was an answer to rapid increase of use and evolution of computer-based technologies, where role of people changed from observing and controlling machinery and equipment to very interactive usage. It was concerned in models of how people perceive, process and use information to achieve their goals. Socio-Technical Systems Design focused on interaction between people and technology in workplaces and organizations, trying to optimize organizational performance and improve humanistic aspects.

Another research area was "Cognitive Modeling". It originated from late 1950s and it's goal is to approximate people's basic cognitive processes and reasoning, explain them and predict how they'll evolve under particular conditions. It first focused on how people solve problem symbolically - taking input in form of symbols, manipulating them and producing output. Later it addressed how information is taken from external world, especially when using computer systems. The approach later evolved in creating models for human-computer interactions, such as Human Model Processor, GOMS (Goals, Operators, Methods and Selection rules), Keystroke Level Models and Programmable User Models.

User Centered Design (UCD) term was coined in 1986 in Norman and Draper work: "User Centered System Design; New Perspectives on Human-Computer Interaction". As Endsley describes in his book [9], UCD was an answer to a trend of "Technology-Centered Design", and key principles were to:

- Focus on user's goals, tasks and abilities
- Organize technology around the way users process information and make decisions
- Keep the user in control and aware of state of the system

A very related movement was that of Human-Centered Design, which considered not only users in interaction with the system, but noticed human capabilities and limitations and the need to understand them. An examples for need for such approach are when the user might not directly interact with the system (as in elderly healthcare monitoring) or how to design systems with consideration for people with social interaction disorders (e.g. autism). Human Centered Design activities are the subject of ISO standard 9241-210:

1. Understanding and specifying the context of use (including users, tasks, environments)
2. Specifying the user requirements in sufficient detail to drive the design
3. Producing design solutions that meet these requirements
4. Conducting user-centered evaluations of these design solutions and modifying the design to take into account the results.

Nowadays the very known popular terms related to this domain are "User Experience" and "Usability". The former is defined as "a person's perceptions and responses that result from the use or anticipated use of product, system, or service" (ISO 1941-210), but also describes person's beliefs and emotions. The latter term is focused on how different tasks in systems are accomplished by the users and what errors do they make in progress - it can be used as interface validation method.

2.3 Startups

Software startups are increasingly popular nowadays and the huge success of companies like Facebook, Spotify, Linkedin and many others as well as very accessible technologies encourage many to create their companies and try to quickly enter new, emerging markets. From software process methodologies point of view, those undertakings are of particular interest, as those young companies operate under time and financial pressure to create the right product for the right market in very uncertain conditions. Some of those companies succeed and many fail - in fact, a recently quoted statistics say that "great majority of such companies fail within two years of their creation" [18] and "more than 90% of startups fail, due primarily to self-destruction rather than competition" [11]. As startups situation differs from established companies, they need a specialized approach.

In fact, to answer this need, Eris Ries proposed the Lean Startup approach and in his 2011 book he defined startup as "a human institution designed to deliver a new product or service under conditions of extreme uncertainty"[21]. Steve Blank, who influenced Eric Ries' ideas highlights 3 key principles of Lean Startup method [5]:

- Instead of creating intricate business plan based on mostly guesswork, summarize hypotheses in a framework called "business model canvas"
- Use "Customer Development" practices (created by Steve Blank) to test hypotheses - create Minimal Viable Product (MVP) to get feedback from customers and then do Build - Measure - Learn cycles, using metrics allowing for verifying learning and then doing "pivots" when hypothesis turns out to be wrong
- Use agile methods to develop Minimal Viable Product. In fact, Lean Startup has it's name after Lean Software Development method, created

in 2003 by Mary and Tom Poppendieck [20]. This approach was adapted from Toyota Production scion-technical system and focused on eliminating waste and amplifying learning.

Furthermore, Steve Blank argues that most common failure factors at startups were the high cost of acquiring first customers, even higher cost of creating the wrong product and too long development cycles. Lean Startup approach, in his opinion, helps to counter those limitations and make startups less risky. What's more, it starts to be taught at universities and is becoming increasingly well adopted in the industry.

2.4 State of the Art

During recent years of "agile methods reigns" there came natural realization that agile methods are not silver bullets as they have limitations due to organizational and development environments in which they are incorporated, particularly they might be limiting for distributed development environments, large teams and large, complex software [25]. Those findings seem to be consistent with Kenneth Rubith, who in his book about Scrum, the most widely adopted agile methodology, compared various domains of decision making (using Cynefin framework) and came to the conclusion that while Scrum is a great fit for "Complex" domain, it's not the best approach for "Complicated", "Chaotic" and "Disorder" domains [23], and the last two are often met in startup environments. However, agile methods can be extended to address their limitations. Paradoxically, those extensions are often in the form of more "traditional" development process approaches [26].

As agile practices are now mainstream and have been there for a while, they are now very well defined and are huge areas and so research related to them has diverged into many branches. Traditionally there are surveys and mapping studies, which try to quantitatively measure the trends and answer questions such as what particular practices are used the most in the industry (time boxing, planning meeting, daily discussions) and what domains are they mostly used in [8]. However, most research activity seems to be at the intersection of agile practices with various domains, most often domains representing more "traditional" approaches.

As an example Chen Yang [27] studied how to combine Software Architecture, that is practices related to high level system design with careful assessment of trade-offs, which was initially critiqued for "Big Design Up-Front" leading to excessive documentation and implementation of unneeded features. But the question remained - "how much design is enough for different classes of problems" and architecture design in early iterations, architecture freeze in later stages and iterative delivery documentation were some of most representative advises found.

Another example is a study about integrating Requirements Engineering. Research about software projects failure rate and factors suggests that among mostly reported causes is too much time spent on rework and creating the

product out of sync with business due to poor requirements engineering. [1]. Requirements Engineering provides in-depth description of practices related to requirements elicitation (interviews, focus groups), analysis (prioritization and modeling), documentation and validation (often with prototyping). Agile Requirements Engineering (RE) differs from traditional RE mostly because of iterativeness - instead of specifying complete specification at the beginning, which most often would be infeasible, requirements are specified iteratively and very often reprioritized (Scrum recommends practice addressing it called "Backlog Grooming" every Sprint[23]). The often found challenge in Agile RE lies in neglecting nonfunctional requirements, because customers are for the most part not preoccupied with technical intricacies developers must face. Often reported causes of problems were ignoring critical aspects like scalability and performance at early stages, which inhibited growth later on. Overall, once claimed as heavyweight, Requirements Engineering have found their place in agile-dominant industry. Researchers emphasize the importance of "intensive communication between the developers and customers as the most important RE practice". [17] [7]

By far most spotlight, however, was directed at the intersection of Agile and User Centered Design communities.

2.4.1 Agile and User Centered Design

Agile is now mainstream, but naturally through it's focus on lightweight approach it doesn't cover all the possible needs. Particularly, Agile community doesn't extensively discuss users or user interfaces, none of most used Agile methodologies includes explicit guidance in the area of usability and there is usually no notion of user interface specialist. Generally, applicability of usability practices in agile systems is considered deficient, with agile methods focusing primarily on iterative deliveries of working features to customers. Meanwhile, the priority of UCD is user satisfaction, and although there exists "philosophical and principled differences between Agile methods and UCD in focus, evaluation method, culture and documentation", there's been a lot of effort in community to define practices using both philosophies and they have been referred to as Agile and User Centered Design Integration (AUCDI) or, most often, simply Agile UX. [24][12]

2.4.1.1 Agile User Centered Design Patterns

AUCDI patterns are described using for each stage of Human Centered Design Process ISO 13407 standard [3][4] :

1. Identify Needs for Human-Centered Design
    - Sprint Zero
    - One Sprint Ahead
    - UX Specialists as Product Owners

- Users time is valuable
- Parallel Tracks
- UX Specialists as Full-Time Member of the Agile Team

2. Specify Context of Use
    - Little Design Up Front
    - Contact Plan of Users
3. Specify Requirements
    - User Stories
    - More collaboration, less documents
    - Prototypes as specification
4. Create Design Solutions
    - Low fidelity prototyping
    - High fidelity prototyping
    - Design Studio
    - Collaborative and Participative Design
5. Evaluate Designs
    - Tests with users
    - Evaluation by inspection
    - RITE method
    - Acceptance tests

All stages of above mentioned patterns should be incorporated for each batch of features in agile iteration. Those patterns are not meant to be used with great formality, their purpose is to share knowledge so teams can select patterns that best fit their specific development environment [4].

2.4.1.2 Usage of Agile UX in industry

Various reports seem to agree that usage of usability practices in surveyed companies have been increasing in recent years. The most established practices in industry seem to be Little Design Up Front (LDUF), Sprint 0, designing one sprint ahead and close collaboration between agile team and UX team. In fact, interaction between agile and UX teams has been reported to be the source of most challenges and pitfalls - namely overworking understaffed UX designers, power struggle with not clearly defined roles of teams, as well as work coordination and communication issues. [24][12].

As agile and UX teams collaboration receives the most attention in various reports, the "parallel interwoven creation tracks", where design team works

ahead of implementation team ("one sprint ahead") and uses artifacts as central means of communication seems to be the most advised approach [6]. Scrum seems to be proposed as the most fitting methodology for incorporating UX practices, as teams can plan together each sprint during Backlog Grooming and can both attend daily Stand-Ups. Moreover, UX member is proposed to be included to Scrum team on daily basis, as part of developer team [16].

There have even been created dedicated framework dedicated to integrating usability engineers into an agile team called Extreme Scenario Based Design Approach (XSBD), with COSBY variant, focusing on explicit usability related metrics, but they have yet to receive broad attention [12].

2.4.2 Startups

Despite the phenomenon in recent years of newly created software companies quickly achieving huge success and receiving wide public attention and recognition, research of this topic is scarce. Startups don't have agreed on definition and they are mostly described through context of their activities, that is extreme uncertainty (as in Eric Ries "Lean Startup" [21] definition), little to no operating history, lack of resources, intensive time pressure, quick time to market and innovativeness. Due to that, reports state that many startups don't use well defined processes, but opportunistically select practices when needed. The same approach is applied to requirements, documentation, architecture and metrics. Managerial practices are reduced to bare minimum and developers are empowered to creatively adapt to several roles. [18].

2.4.2.1 Lean UX

3 Problem formulation and methodology

Case study.

4 DNCR project summary

5 Solution evaluation

6 Conclusion

References

[1] Giuseppe Arcidiacono. Comparative research about high failure rate of it projects and opportunities to improve. *PM World Journal*, 6(II), 2017.

[2] Kent Beck, Mike Beedle, Arie Van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, et al. The agile manifesto, 2001.

[3] Ana Paula O Bertholdo, Tiago Silva da Silva, Claudia de O Melo, Fabio Kon, and Milene Selbach Silveira. Agile usability patterns for ucd early stages. In *International Conference of Design, User Experience, and Usability*, pages 33–44. Springer, 2014.

[4] Ana Paula O Bertholdo, Fabio Kon, and Marco Aur'elio Gerosa. Agile usability patterns for user-centered design final stages. In *International Conference on Human-Computer Interaction*, pages 433–444. Springer, 2016.

[5] Steve Blank. Why the lean start-up changes everything. *Harvard business review*, 91(5):63–72, 2013.

[6] Manuel Brhel, Hendrik Meth, Alexander Maedche, and Karl Werder. Exploring principles of user-centered agile software development: A literature review. *Information and Software Technology*, 61:163–181, 2015.

[7] Lan Cao and Balasubramaniam Ramesh. Agile requirements engineering practices: An empirical study. *IEEE software*, 25(1), 2008.

[8] Philipp Diebold and Marc Dahlem. Agile practices in practice: a mapping study. In *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, page 30. ACM, 2014.

[9] Mica R Endsley. *Designing for situation awareness: An approach to user-centered design*. CRC press, 2016.

[10] Alfonso Fuggetta and Elisabetta Di Nitto. Software process. In *Proceedings of the on Future of Software Engineering*, pages 1–12. ACM, 2014.

[11] Carmine Giardino, Xiaofeng Wang, and Pekka Abrahamsson. Why early-stage software startups fail: a behavioral framework. In *International Conference of Software Business*, pages 27–41. Springer, 2014.

[12] Gabriela Jurca, Theodore D Hellmann, and Frank Maurer. Integrating agile and user-centered design: a systematic mapping and review of evaluation and validation studies of agile-ux. In *Agile Conference (AGILE), 2014*, pages 24–32. IEEE, 2014.

[13] Rupinder Kaur and Jyotsna Sengupta. Software process models and analysis on failure of software development projects. *arXiv preprint arXiv:1306.1068*, 2013.

[14] Craig Larman and Victor R Basili. Iterative and incremental developments. a brief history. *Computer*, 36(6):47–56, 2003.

[15] Version One. 11th annual state of agile development survey, 2017.

[16] Tina Øvad and Lars Bo Larsen. The prevalence of ux design in agile development processes in industry. In *Agile Conference (AGILE), 2015*, pages 40–49. IEEE, 2015.

[17] Frauke Paetsch, Armin Eberlein, and Frank Maurer. Requirements engineering and agile software development. In *Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003. WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on*, pages 308–313. IEEE, 2003.

[18] Nicolò Paternoster, Carmine Giardino, Michael Unterkalmsteiner, Tony Gorschek, and Pekka Abrahamsson. Software development in startup companies: A systematic mapping study. *Information and Software Technology*, 56(10):1200–1218, 2014.

[19] PMI. Pmi's pulse of the profession 2017, 2017.

[20] Mary Poppendieck and Tom Poppendieck. *Lean Software Development: An Agile Toolkit: An Agile Toolkit*. Addison-Wesley, 2003.

[21] Eric Ries. *The lean startup: How today's entrepreneurs use continuous innovation to create radically successful businesses*. Crown Business, 2011.

[22] Frank E Ritter, Gordon D Baxter, and Elizabeth F Churchill. User-centered systems design: a brief history. In *Foundations for designing user-centered systems*, pages 33–54. Springer, 2014.

[23] Kenneth S Rubin. *Essential Scrum: A practical guide to the most popular Agile process*. Addison-Wesley, 2012.

[24] Dina Salah, Richard F Paige, and Paul Cairns. A systematic literature review for agile development processes and user centred design integration. In *Proceedings of the 18th international conference on evaluation and assessment in software engineering*, page 5. ACM, 2014.

[25] Dan Turk, Robert France, and Bernhard Rumpe. Assumptions underlying agile software development processes. *arXiv preprint arXiv:1409.6610*, 2014.

[26] Dan Turk, Robert France, and Bernhard Rumpe. Limitations of agile software processes. *arXiv preprint arXiv:1409.6600*, 2014.

[27] Chen Yang, Peng Liang, and Paris Avgeriou. A systematic mapping study on the combination of software architecture and agile development. *Journal of Systems and Software*, 111:157–184, 2016.

[28] Edward Yourdon. Death march: The complete software developers guide to surviving mission impossible projects. paul, db, 1997.