Generative patterns for designing multiple user interfaces

Conference Paper · May 2016

DOI: 10.1145/2897073.2897084

3 authors:

Thanh-Diane Nguyen
Université Catholique de Louvain - UCLouvain
3 PUBLICATIONS 14 CITATIONS
SEE PROFILE

Jean Vanderdonckt
Université Catholique de Louvain - UCLouvain
487 PUBLICATIONS 7,447 CITATIONS
SEE PROFILE

Ahmed Seffah
Lappeenranta – Lahti University of Technology LUT
204 PUBLICATIONS 2,845 CITATIONS
SEE PROFILE

Generative Patterns for Designing Multiple User Interfaces

Thanh-Diane Nguyen, Jean Vanderdonckt
Louvain School of Management,
Université catholique de Louvain
Place des Doyens, 1 - 1348 Louvain-la-Neuve (Belgium)
{thanh-diane.nguyen,jean.vanderdonckt}@uclouvain.be

Ahmed Seffah
Innovation and Software, School of Business and Management, Lappeenranta University of Technology
P.O.Box 20, FI-53851 Lappeenranta (Finland)
ahmed.seffah@lut.fi

ABSTRACT

End users interacting with mobile services through a wide diversity of mobile devices and platforms inevitably endure various user experiences when no consistency is ensured across these devices and platforms. Developing the same service for heterogeneous devices remains a challenging task: how to ensure that the service will provide end users with the same level of user experience or at least a common minimum level of usability across software development and deployment platforms. This paper addresses this problem by introducing a generative design pattern-based approach for cross-device services: a design pattern captures frequent interactive behaviors at a higher level of abstraction than the code level, the selection of such a design pattern is then subject to parametrization so as to drive a code generation process. The pattern is not only considered descriptive, since it specifies a consistent user experience across devices, but also generative because it ensures some consistency across different devices and platforms since the pattern is instantiated in the same way for each device or platform. To exemplify this process, the master-details design pattern is detailed and illustrated on a case study for a car rental mobile service. A pilot study conducted with thirty-two participants suggests that this is a viable approach for quickly producing multi-devices services, with limited development effort, but also with limited variability.

CCS Concepts

• **Human-centered computing → Ubiquitous and mobile computing → Ubiquitous and mobile devices • Software and its engineering → Software creation and management → Designing software**

Keywords

Cross-device pattern; Generative pattern; Mobile service development; Multiple User Interfaces; User interface pattern.

1. INTRODUCTION

Today's wide variety of computing devices and platforms enable end users to carry out their interactive task in virtually any context of use: any time, any space, with any device or platform [16,31,32]. *Multiple User Interfaces* (MUIs) refer to a user interface that provides access to the same information or service using multiple computing devices or platforms while providing the end user with a similar user experience [7]. As opposed to cross-device user interfaces [23] which are intended to produce different, possibility inconsistent, user interfaces for different devices or platforms, MUIs are aimed at maximizing the uniformity of the user experience across these devices or platforms, even at the price of not exploiting the full potential of interaction capabilities of the target device or platform. This results into *Multi-User Experience (MUX)* [17], which is aimed at delivering a similar user experience through various devices or platforms, e.g., by consistency, but not only. Developing a single user interface for a service running on a specific device with a particular operating system is common, whereas MUI engineering leads developers to consider several challenges [32]:

- MUIs should be compliant with the various style guides [3] recommended by the different operating systems such as Android [2] and Apple [20] or even specialized ones like Garmin when the service need to be accessed via other devices such as a GPS.
- MUI should take into account the available interaction capabilities [5,11], while taking into account the respective constraints imposed by each device or platform, such as: screen size, screen resolution, interaction modalities, and processing capabilities.
- MUI should also adhere to an appropriate architectural and programming language style [5].
- MUI should favor a global multi-device experience [17] induced by using multiple user interfaces as opposed to the sum of the local user experiences induced by each individual user interface.

To address these challenges, some approaches exist such as:

- *S1: Insourcing*: hiring a significant team of developers in which all concerned devices and platforms are covered by a sufficient expertise handle specific platform development including programing languages and best development practices, while having skills for delivering a reasonable user experience for each of them [5].
- *S2: Outsourcing*: devices and platforms for which little or no expertise are present in the development team are covered by third party companies benefiting from such an expertise [16].
- *S3: Responsive design*: all concerned devices and platforms are covered by expertise expanded from one device to another so as to ensure cross-device development [21,32].
- *S4: Model-based design*: adopting a model-based approach [12] or a model-driven engineering approach where a computing-independent model is transformed into a platform-independent model, which is in turn transformed into a platform specific one to generate code that is appropriate for a particular device.

Each aforementioned approach has some advantages and drawbacks and no ideal solution exists. S1 probably delivers the most accurate and powerful output of the software development life cycle, but is the most expensive. S3 attempts to factor out the common ground of all devices through which the mobile service should be provided, with little variations for individual devices. As opposed to S3, S4 attempts to capture the MUI design through a series of abstractions specified in MUI conceptual models [3;26], but developers need to be trained to user interface modeling for this purpose and not all MUIs could be produced in this way. Even if this is the case, not all possible MUIs should be developed in the same way, which may lead to prohibitive resource expenses in development.

MobileSoft'16, May 16-17, 2016, Austin, TX, USA
© 2016 ACM. ISBN 978-1-4503-4178-3/16/05…$15.00
DOI: http://dx.doi.org/10.1145/2897073.2897084

Rather than maximizing the advantages of any of these solutions, this paper is aimed at concentrating on some of these advantages and combining them into a framework for the creation, the development, and the management of cross-device mobile services based on generative patterns in MUIs context. The concept of generative design pattern is exploited here: *i)* to capture MUI design experience into a proved MUI, *ii)* to build a MUI common ground so that the MUI will deliver the same user experience throughout all devices/platforms (as opposed to a code generation that is targeted for each device/platform), and *iii)* to be used as building block in the construction, transformation and code generation. To achieve this goal, the contributions of this paper are:

- **MAD (MAster Detail) Framework** (primary): a framework for designing MUIs for cross-device services based on seven adaptation dimensions (i.e., when, to what, who, where, how, what, why to design) and structured according to the four levels of abstraction defined in the Cameleon Reference Framework (CRF) [7] and suggested by W3C [22].
- **MAD pattern** (secondary): the application of the above framework to one particular type of design pattern, i.e. the master-and-details pattern found in a domain model, along with MUI design options structured according to the four levels of abstraction. A discussion is also provided that explains how to perform a similar work for other types of patterns.
- **UsiMAD application** (tertiary): a software tool that supports the previous M-D pattern by providing the designer with MUI design options so as to generate one or many MUIs for the same pattern applied to a domain. A "car rental" case study exemplifies how UsiMAD could be used.
- **UsiMAD validation** (quaternary): a pilot study has been conducted with participants in order to collect their opinion on using UsiMAD in their development context.

This paper is divided into the following sections: Section 2 presents some related work, Section 3 explains the computational framework, its components and its support, Section 4 applies the framework to the master-details design pattern on a case study, Section 5 presents the validation based on experiments of this tool and Section 6 concludes the paper and presents some avenue of this work.

2. RELATED WORK

2.1 Design Patterns

The wide variation of new devices and platforms brings a new dynamic in UI development [16,17,31] and a considerable heterogeneity of interaction contexts (users, devices and environments) [7, 32]. Hence, MUIs have to adapt to any significant variation of this context of use, not all variations. Since Christopher Alexander introduced design patterns in 1977 "to clearly and succinctly describe particular workplaces, in order to understand possible impacts of new technologies" [1], several *Human-Computer Interaction* (HCI) or user interface pattern collections were introduced in order to capture the best practices of user interface design and to reproduce them in similar contexts [10,27]. These collections include representative examples such as, but not limited to: Henninger' collection of guidelines and design patterns along with a learning method [14], design patterns for mobile platforms [25], conceptual patterns for the OlivaNova environment [26], Tidwell descriptive patterns [28], van Duyne *et al.* e-commerce patterns [29], and van Welie design patterns [30]. Most of these collections contain only *descriptive patterns*, i.e., patterns described with a various level of details, but not direct way to put them into practice or encapsulate them into an *Interactive Development Environment* (IDE) or in any particular software development method.

2.2 Applications of Design Patterns

Some HCI collections even cover design patterns for different mobile services, devices, platforms and operating systems [9,31] by providing different examples depending on the context of use. A major concern with them is that they do not provide any methodological guidance on applying these patterns or any development method to develop the related code. In order to address the aforementioned challenges in the MUI context, requirements [33] have been elicited towards *generative patterns* where patterns are augmented with code generation capabilities to immediately apply a pattern when needed and generate the corresponding code.

GUIDE (Guidelines for Usability through Interface Development Experiences) [14] provides an interactive environment in which usability guidelines and design patterns are presented to the designer, along with information on when, where to apply them. However, the 'how' dimension was left to the developer's appreciation.

PaGIMS [10] introduced a first form of software support towards applying design patterns by relying on a user interface model: instead of applying a design pattern manually (this process is therefore error prone since depending on the designer's or developer's expertise and rigorousness), a user interface model is specified in which the pattern is applied. PaGIMS does not support MUIs.

Conceptual patterns [26] have been introduced in the OO-Method in order to produce user interfaces for three platforms/languages: HTML, C++ for Visual Studio, and JavaBeans. In this environment, the designer picks the design pattern she wishes to apply among a collection of predefined patterns (e.g., filter, navigation), combines the patterns together, and let the software generate the code corresponding to the desired targets. Only 3 possible user interfaces could be generated for each design pattern (one for each language), but consistency is preserved since the presentation and the navigation is consistent from one platform to another, although some usability study shows some significant differences [3].

2.3 Master-And-Details Pattern

Let us to focus on one particular design pattern: the *Master-And-Details* (MAD) pattern [24], which displays a master list (e.g., a list of cars) and then details any currently selected item (e.g., a particular car). This pattern has been selected for the following reasons: it starts from a domain model, thus offering a data oriented perspective and a conceptual starting point [26]; it is widely used both in the literature and in practice, by designers, developers, by private ones and software vendors; it is largely considered in systematic development of interactive information systems [9]; previous works do not evaluate the heterogeneous device dimension of this pattern in the light of UI implementation and usability concerns [3]. In order to compare how the MAD pattern is implemented in existing works, we will rely on the Cameleon Reference Framework [7], which is suggested by W3C for structuring model-based design of UIs [22]. Similarly to the *Model-Driven Engineering* (MDE) perspective, this framework consists of four levels of abstractions: Task model and Domain model, Abstract User Interface (AUI), Concrete User Interface (CUI), Final User Interface (FUI) [7]. Other CRF-compliant frameworks exist [22].

Table 1 compares entries in the related work and general-purpose software and IDEs with respect to how they support the master-and-details pattern according to the four CRF levels. Regarding the task and domain level, toolkits and IDEs are primarily restricted in that they only support some data or domain model, explicitly or implicitly, but no task model. Other works have some task model whose representation ranges from implicit [26,28] to explicit [10].

	Task and domain	AUI	CUI	FUI
GUI toolkit	◑	○	○	●
Objective-C [20]	○	○	○	●
Android IDE [2]	◔	○	○	●
Visual Studio	◔	○	○	●
Odoo	○	○	○	●
Oracle	◔	○	○	●
Balsamiq [4]	○	○	○	◑
PaGims [9,10]	◑	◔	◔	●
Quill [13]	◕	◕	◕	●
Guide [12]	◔	○	○	○
Nebeling [20]	◑	○	◕	●
Nilsson [21]	◔	○	◑	●
OlivaNova [22]	◔	○	◔	●
Tidwell [25]	◔	○	○	◔
van Duyne *et al.* [26]	◔	○	○	◔
van Welie [27]	○	○	○	◔
MAD [this paper]	●	●	●	●

Table 1. Level of support of the master-and-details patterns expressed according to Harvey's Balls

The AUI level is almost unsupported, apart partially in PaGims [9] and explicitly in Quill [11], but the AUI model is not linked to the previous level. The CUI level is more supported than the AUI level, but it highly depends of the global approach adopted in the work. For instance, PaGIMS [9], Quill [11], and OlivaNova [22] do support CUI, but only for graphical user interfaces, even if this type of graphical user interface is supposed to be responsive. Nebeling *et al.* [20] probably provide the richest support at this level since it consider different interaction modalities for cross-devices user interfaces. Regarding the FUI level, almost all entries have some level of support, apart from purely descriptive patterns or patterns containing only some code fragments (e.g., [25,26,27]).

From Table 1, we could conclude that no existing work today covers equally the four CRF levels of abstraction via user interface design patterns. To overcome this limitation, the following section reviews and extends the definition of MAD pattern from previous work [24] in a specific framework focused on cross-devices services. To support the usage of user interface design patterns defined in this context, a case study will be used as a running example.

3. MAD FRAMEWORK

3.1 The MAD Domain Model

In a Master-And-Details pattern, a master is displayed consisting of a series of items for which details cold be conveyed on demand. A generic domain model of the MAD pattern could represented by a UML 2.0 Class Diagram (Fig. 1), in which the master is characterized by its own attributes (i.e. typical attributes like ID, Name, etc. and specific master attributes) and its own methods (i.e., typical methods derived from the Create-Read-Update-Delete-Search [CRUDS] pattern [6] and specific master methods). Similarly, any item of the master collection is characterized by its own attributes (i.e. typical attributes and specific item attributes) and methods (i.e. typical methods and specific item methods).

3.2 The MAD Task Model

Based on a generic domain model (Section 3.1), a *task model* should then be created in order to describe how the end user could carry out her interactive task in a way that is independent of any context of use and independent of any implementation. For these reasons, a task model expresses how an end user is conducting a task without referring to any particular system implementation.

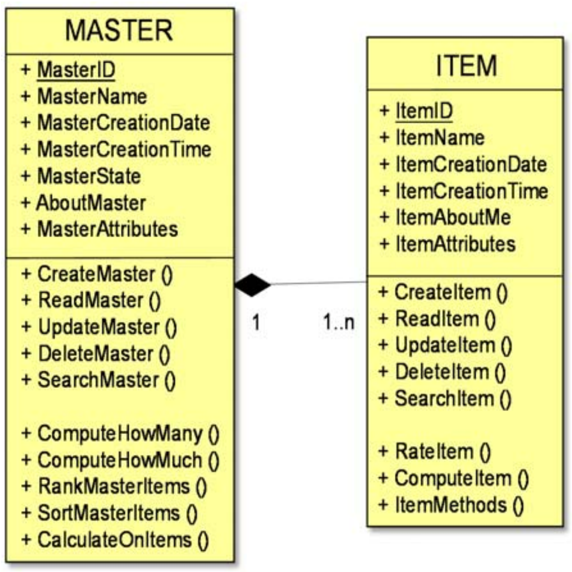


Figure 1. A UML Class Diagram of the domain model.

In the same way a generic domain model is produced, a generic task model is derived based on possible interactions between the master and any item as follows (Fig. 2):

1. *Interacting with the master*: the user can manage the master class by editing any master attribute and/or triggering any master method, including selecting a particular item.
2. *Interacting with an item from the master*: the user can manage any item by editing any item attribute and/or executing any item method.

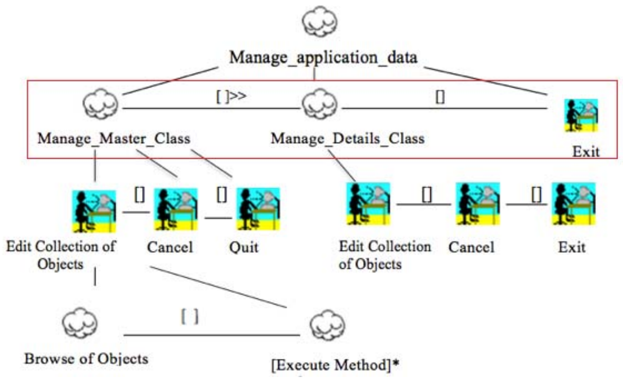


Figure 2. A generic MAD task model.
Legend: T1 [ ]>> T2 represents a sequence with information passing between task T1 and T2, T1[]T2 represents a choice between T1 and T2, T1* represents an iterated task.

4. THE USIMAD APPLICATION

In order to exemplify how to proceed with the rest of the generative pattern-based approach, let us consider a case study for car rental. In this scenario, a pool of cars is available from which a person, having some identity and preferences for cars, is looking for potential cars. A partial domain model is reproduced in Fig. 3, which is an instantiation of the generic domain model from Fig. 1. The generic task model of Fig. 2 remains valid and is simply instantiated on the particular domain model of Fig. 3. In this way, a car could be added, removed, updated from the car pool, a car could be retrieved and searched against some search criteria (e.g., category, color, possible options). The transition from the Task & Domain level to the AUI level, from AUI to CUI, and from CUI to FUI are respectively explained and illustrated in the next sub-sections.

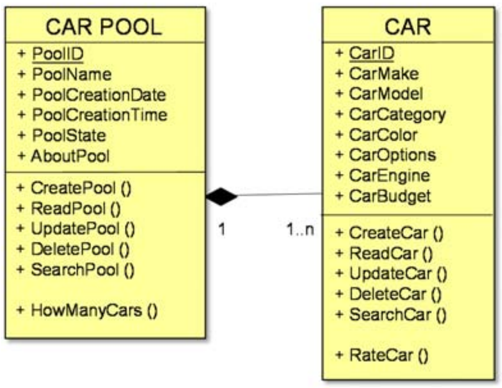


Figure 3. A partial domain model for car rental.

4.1 From task and domain models to abstract user interface

Once the domain model is encoded via a domain model editor which stores the classes and their relationships, since the task model is already pre-defined, the next step consists of structuring an abstract user interface in terms of so-called *Abstract Interaction Units* (AIUs) [22], which could be recursively decomposed in sub-units. An AIU is defined as a part of the task model that will be conveyed at once when the end user will interact with the system. Fig. 4 reproduces a screenshot of the AIU editor in which the domain model is imported. The designer is then free to edit AUI in terms of groups (AIUs and sub-AIUs) by drag and drop. Fig. 4 depicts an abstract user interface decomposed into 4 AIUs (i.e. identity, address, preferences for a car, and comments). The "Preferences" AIU imported some attributes from the CAR class, while the "Identity" and the "Address" AIUs did the same from a Customer class.

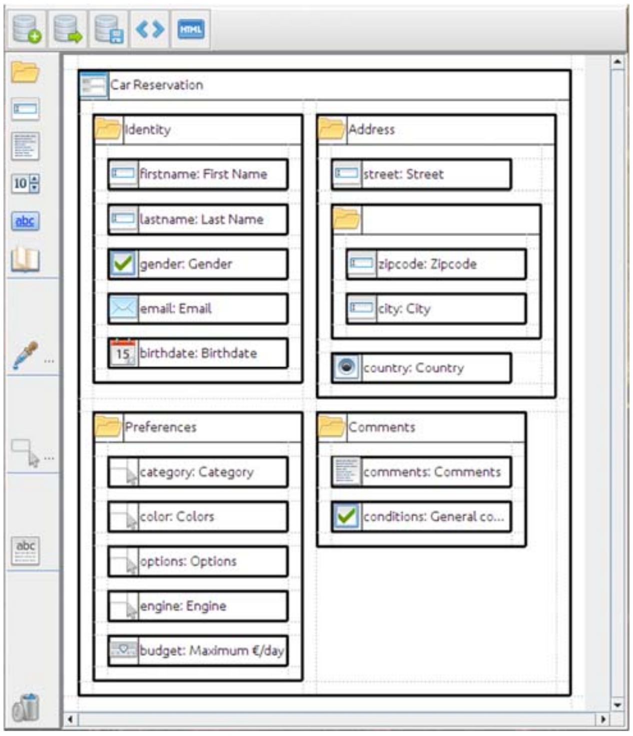


Figure 4. A car rental Abstract User Interface.

4.2 From abstract to concrete user interface

At the AUI level, only a hierarchy of abstract interaction units is defined without any reference to any interaction modality. Let us assume that the graphical modality will be used from now on since it is the most frequently used one. Other interaction modalities, like for vocal, tactile, touchable interfaces, could be elaborated similarly, but multimodal MUIs is beyond the scope of this paper.

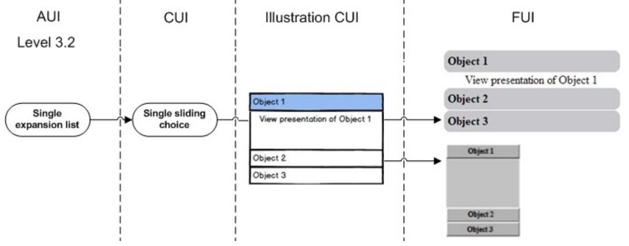


Figure 5. Choosing design options for MAD in UsiMAD.

In order to ensure the transition from abstract user interface (no modality known) to concrete user interface (modality known, but independent from any implementation), UsiMAD[1] enables the designer to choose how the master-and-details pattern will be presented in a future graphical user interface, here from the car pool to a particular car belonging to this pool. The designer will decide the values of high-level design options by picking one possible value at each level of UsiMAD's decision tree (Fig. 5, see Appendix for the complete decision tree implemented) that affects the potential values of the sub-sequent levels.

First, the amount of items to be displayed simultaneously should be decided (Fig. 6): one item of the collection at a time (i.e., cars one by one), many items in a group (i.e., a partial list of cars), or all items resulting from the search at once (i.e., a complete list of cars). Then, depending on this choice, UsiMAD's decision tree directs the designer to a new question: how attributes and methods of the item should be conveyed, here in a single expansion list, and so forth. Depending on the specifications of the user and the device/platform, UsiMAD suggests an appropriate value based on usability knowledge available until a final node of the decision tree is reached. Yet, in order to avoid any reference to any particular target, the various design options are expressed in high level terms. For instance, a tabbed list instead of a tabbed dialog box. Fig. 7 reproduces two final user interfaces, one for a smartphone and one for a tablet.

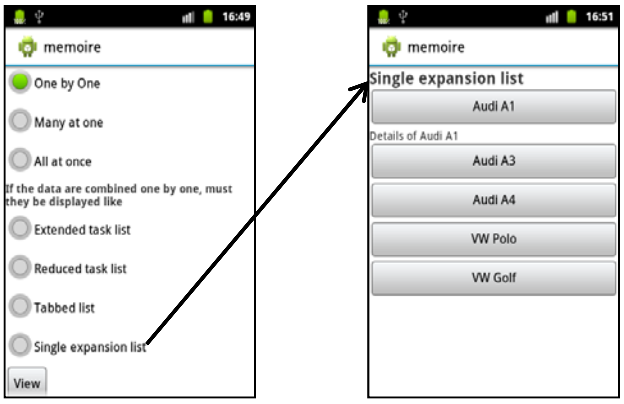


Figure 6. Choosing design options for MAD in UsiMAD.

[1] See http://usimad.alwaysdata.net/tree.

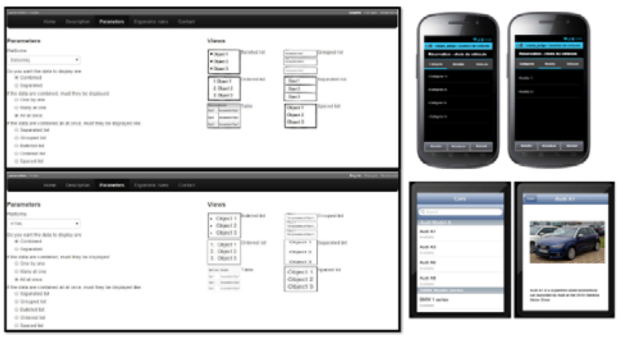


Figure 7. UsiMAD final choice for a smartphone and a tablet.

If the designer is willing to preserve a uniform user experience across devices, it is recommended to consistently adopt the same values for high-level each design option for all devices. Regarding low-level options, different values could be chosen without affecting the global user experience. If the designer wishes to understand the rationale beyond the suggested value, a list of usability guidelines that are respected by construction is displayed by UsiMAD. Any particular usability guideline that is no longer satisfied by a particular choice becomes unchecked. Understanding the impact of each value of each design option is progressive when using UsiMAD. To foster this understanding, UsiMAD at run-time provides designers with a *user interface generation preview* at both CUI and FUI levels:

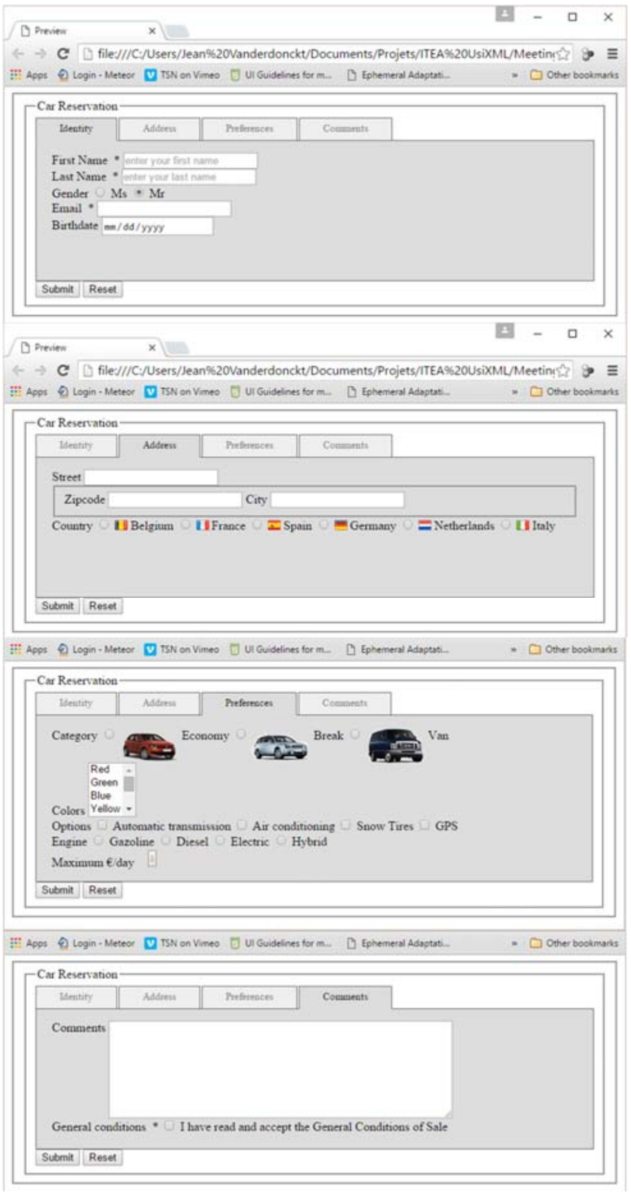
- At *CUI level*, a Balsamiq [4] mock-up is presented corresponding to the general domain model. This mock-up has been predefined and does not rely on the actual domain model. This is why a second preview is offered, which is described below.
- At *FUI level*, a UsiXML file is generated on-the-fly based on the abstract user interface and its decided options that lead to a semi-automated generation of a resulting final UI. For the moment, three targets are available: HTML5 (for instance, Fig. 8 depicts the resulting graphical user interface for desktop, while a responsive HTML5 [18] version is also available), Android [2] (right part of Fig. 6 and Fig. 9 partially depict this situation), and Objective-C [20].

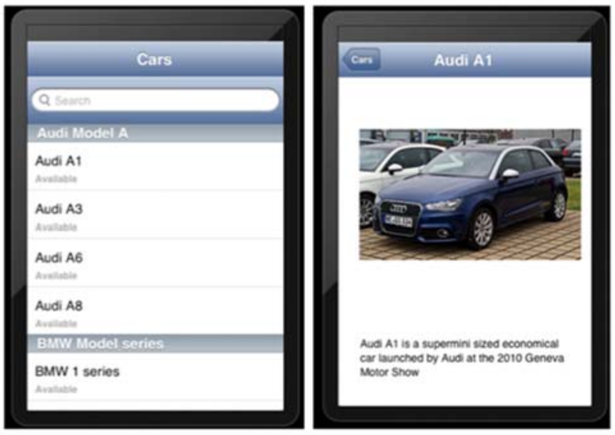
4.3 From concrete to final user interface

Once design options have been decided (note that default values are automatically assigned by UsiMAD), a transition from CUI level to FUI could be achieved. The cross-device pattern is available for three targets: in HTML5 for both desktop (Fig. 8) and mobile, in a responsive version, in Objective-C for iOS-based devices (Fig. 9), and in Android IDE for Android smartphones and tablets (Fig. 10).

A *Model Voyager* (http://sites.uclouvain.be/mbui/) illustrates how to navigate through the 4 levels of abstraction of the Cameleon Reference Framework starting from a same task model and for a predefined domain model. In this on-line version, multiple user interfaces have been produced for different targets, depending on various users and devices. Starting from the same task and domain models, it is possible to explore various design options level by level or jump from one level to another while keeping constant some variable of the context of use (e.g., the user is the same).

Any project for a particular case could be added, edited, or removed from this library. Another library could be created for another problem, with different task and domain models. Each time a branch is expanded, possible values for decided design options are visualized and the branch is further expanded to the subsequent levels, thus facilitating how decided design options may affect the resulting final user interface.


Preview
file:///C:/Users/Jean%20Vanderdonckt/Documents/Projets/ITEA%20UsiXML/Meeting
Apps Login - Meteor TSN on Vimeo UI Guidelines for m... Ephemeral Adaptati... Other bookmarks
Car Reservation
Identity Address Preferences Comments
First Name * enter your first name
Last Name * enter your last name
Gender Ms Mr
Email *
Birthdate mm/dd/yyyy
Submit Reset
Street
Zipcode City
Country Belgium France Spain Germany Netherlands Italy
Category Economy Break Van
Red Green Blue Yellow
Colors
Options Automatic transmission Air conditioning Snow Tires GPS
Engine Gazoline Diesel Electric Hybrid
Maximum €/day
Comments
General conditions * I have read and accept the General Conditions of Sale
Figure 8. HTML5 for desktop final user interface.



Audi A1 is a supermini sized economical car launched by Audi at the 2010 Geneva Motor Show

Figure 9. Objective-C iOS final user interface.

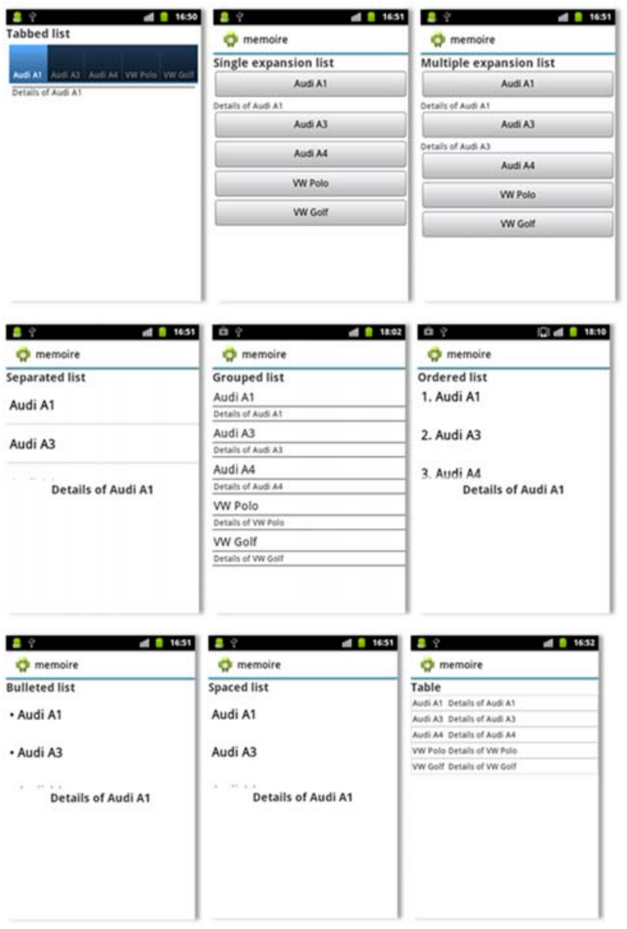


Fig. 10. Android for mobile final user interface.

4.4 Other multi-device design patterns

Based on the UsiMAD framework, a cross-device design pattern is hereby defined as a template of objects with stereotypical responsibilities and interactions; the template may be applied by analogy on any other domain model, but with the same stereotypical task model. The main advantage of this approach is that there is no need to produce any task model anymore since it is pre-defined and it is used consistently from one device to another, thus producing by construction a uniform user experience from one device to another and achieving some form of multi-device experience [17,31]. The main drawback of this approach is that this simplification is at the price of having no flexibility in the task model.

While the master-and-details pattern is probably one of the most frequently used pattern in information systems, it is certainly not the sole one. Based on Coad, North & Mayfield design patterns [8], five families of design patterns are currently under investigation: *i) fundamental patterns* (e.g., beyond the MAD, filter, navigation), *ii) transaction patterns* (e.g., Actor-Participant, Participant-Transaction, Place-Transaction, Specific Item - Transaction), *iii) aggregate patterns* (e.g., Container-Content, Container - Container Line Item, Group-Member, Assembly-Part), *iv) plan patterns* (e.g., Plan-Step, Plan-Plan execution, Step-Step execution), and *v) interaction patterns* (e.g., Peer-to-Peer, Proxy-Specific Item, Publisher-Subscriber). Note that some of these more specific patterns could be considered themselves as a particular application of the MAD pattern with some structure.

When one of these design patterns is itself an instantiation of a fundamental pattern, the corresponding cross-device user interface pattern is derived by restriction of the potential values. For instance, a Plan-Step pattern could be considered as a particular case of a master-and-details pattern where Steps are presented sequentially, on a time-based manner. Therefore, the potential values for the design options are no longer the full range of potential values offered by the master-and-details pattern, but adapted in a subset, for instance, a slider, a time-based scroll bar, a wizard of options. Based on the OlivaNova [26], other conceptual patterns are also considered for extension of UsiMAD. These patterns include: population interaction unit, service interaction unit, filter, etc. These patterns are available for three targets only: HTML, C++ for Visual Studio, and JavaBeans, all of them for desktop. It makes sense to expand the range of these patterns to other devices/platforms in the same way.

5. PILOT STUDY

In order to assess the impact of the MAD framework supported by UsiMAD, a pilot study has been conducted stated as a Goal/Question/Metric (GQM) paradigm [6]: *Analyze* the generative design approach (object of study) *for the purpose of* evaluation (purpose) *with respect to* the usability of the MAD framework supported by UsiMAD (quality focus) *from the point of view* of the designer (point of view) *in the environment of* the pattern (environment).

5.1 Method

Participants. We conducted a user trial of 32 users (7 female, 25 male) who were recruited from a database of volunteers and a student mailing list coming from computer-related disciplines (e.g., design, development, testing) and having different ages and backgrounds. Participants were invited via e-mail to participate in the experiment: an e-mail was sent to them along with an introduction text about the problem (scenario), instructions about the task to be performed with UsiMAD, a link to UsiMAD, and a form to fill-in the IBM PSSUQ (Post-Study System Usability Questionnaire) questionnaire. This questionnaire consists of two series of questions: 19 questions inherited from the IBM CSUQ (Computer Satisfaction Usability Questionnaire) [18] and 7 questions related to the particular assessment of concern (Fig. 11). For each question of the closed questionnaire, we asked the participants to respond to a series of positive statements on a 7-point Likert scale [19] of one to five (1=strongly disagree, seven=strongly agree). The IBM PSSUQ questionnaire has been selected because it has been empirically validated and it benefits from a high statistical correlation of 0.89 (Alpha coefficient) with respect to usability of the tested system. For these reasons, the original questions as stated in the IBM CSUQ are left unchanged in order to preserve this empirical validation. Each question falls in one category, the first 19 ones coming from CSSUQ [18], questions 20 to 26 being added for the PSSUQ:

1. System usefulness (SYSUSE: Items 1-8)
2. Quality of the information (INFOQUAL: Items 9-15)
3. Quality of the interaction (INTERQUAL: Items 16-18)
4. Overall quality of the system (OVERALL Items 19, 25,26)
5. Usability (USABILITY: Items 20-24)

The newly added questions on top of the 19 CSUQ questions were: *Q20. I always know where I am and how to go where I want.* This question was required because UsiMad navigates within a decision tree, with questions on each node and new branches being expanded depending on the answer to each question.

Q21. Returns and outputs are explained and visible. Using abstract terms in the 4 CRF levels could be interpreted as challenging by participants who were unfamiliar with user interface design. Therefore, we need to know whether the information was visible.

Q1.	Overall, I am satisfied with how easy it is to use this system
Q2.	It was simple to use this system
Q3.	I can effectively complete my work using this system
Q4.	I am able to complete my work quickly using this system
Q5.	I am able to efficiently complete my work using this system
Q6.	I feel comfortable using this system
Q7.	It was easy to learn to use this system
Q8.	I believe I became productive quickly using this system
Q9.	The system gives error messages that clearly tell me how to fix problems
Q10.	Whenever I make a mistake using the system, I recover easily and quickly
Q11.	The information (such as online help, on-screen messages, and other documentation) provided with this system is clear
Q12.	It is easy to find the information I needed
Q13.	The information provided for the system is easy to understand
Q14.	The information is effective in helping me complete the tasks and scenarios
Q15.	The organization of information on the system screens is clear
Q16.	The interface of this system is pleasant
Q17.	I like using the interface of this system
Q18.	This system has all the functions and capabilities I expect it to have
Q19.	Overall, I am satisfied with this system
Q20.	I always know where I am and how to go where I want
Q21.	Returns and outputs are explained and visible
Q22.	The address of the current page lets me know where I am
Q23.	The caption or text allows me to understand the meaning of each part of the tool
Q24.	Information is easily readable
Q25.	Overall, I am satisfied by the guide by parameters
Q26.	Overall, I am satisfied with the guidance offered by usability guidelines

Figure 11. The PSSUQ questionnaire used in the pilot study.

Q22. The address of the current page lets me know where I am. This question elaborates on question 1 about the navigation and to know if displayed information is visible enough.

Q23. The caption or text allows me to understand the meaning of each part of the tool. This question elaborates on question 2 in that it is focusing on elements displayed in the UsiMAD software.

Q24. Information is easily readable. Using abstract terms, it is necessary to know whether participants can read information easily enough to guarantee an appropriate access to information.

Q25. Overall, I am satisfied by the guide by parameters. We evaluate the global subjective satisfaction of participants working with generative patterns by playing with design options.

Q26. Overall, I am satisfied with the guidance offered by usability guidelines. This question was intended to assess whether participates value to be guided based on affective usability knowledge.

Scenario. Each participant to the experiment had to design a user interface for a car dealer to be accessible by two users (i.e., novice versus experts) from three devices (i.e., a smartphone, a tablet, and a personal computer). Participants were instructed to use UsiMad to design such a MUI according to the MAD framework explained in the tutorial. At the end of their design period, they had to generate a UsiXML corresponding to the intended MUI and they had to complete the PSSUQ questionnaire as stated before.

5.2 Results and Discussion

The participant population was distributed as reported in Table 2. The cumulated histogram in Figure 11 summarizes the responses to the questions raised in the PSSUQ questionnaire. *Results from the CSUQ part.* A general observation of Fig. 12 suggests that the subjective satisfaction of participants involved in the experiment follows a rather positive trend. Each cumulated horizontal histogram of Fig. 12 could be interpreted as follows: a score between 6 and 7 is considered as excellent, a score of 5: good, a score of 4 is considered average, a score of 3 is considered poor, and a score between 1 and 2 is considered very bad. In general, a score between 'average' and 'excellent' should not raise any particular concern regarding the question, whereas a score between 'poor" and 'very bad' should raise some discussion. In particular, "Q1. Overall, I am satisfied with how easy it is to use this system" (μ=5.81, M=6, σ=0.72) suggests that the general attitude of participants regarding the whole process supported by UsiMAD could be interpreted as satisfying with respect to the goal: no 'very bad' score has been given by nobody, which is also the case of "Q13. The information provided for the system is easy to understand" (μ=5.59, M=6, σ=0.79), "Q15. The organization of information on the system screens is clear" (μ=5.31, M=5, σ=0.81), "Q19. Overall, I am satisfied with this system" (μ=5.38, M=5, σ=0.80), and "Q26. Overall, I am satisfied with the guidance offered by usability guidelines" (μ=5.34, M=6, σ=0.80). Surprisingly, Q15 and 19, both general purpose questions, are the only questions having a median of 5 from the whole questionnaire, as opposed to 6 for all others.

This is also confirmed by the relatively acceptable scores aggregated in the 5 categories from the CSSUQ questionnaire. Indeed, the results of the presented analysis show that the average score for all categories is between 5 and 7. Overall, participants did appreciate the quality of information and interaction they have had.

Gender	Female:22% Male: 78%	**Computer usage in professional job**	Yes:59% No:41%
Study grade	Bachelor:34% University:66%	**Age**	<25 year:78% [26..35 year]:18% >36 year: 3%

Table 2. The participant population of the experiment.

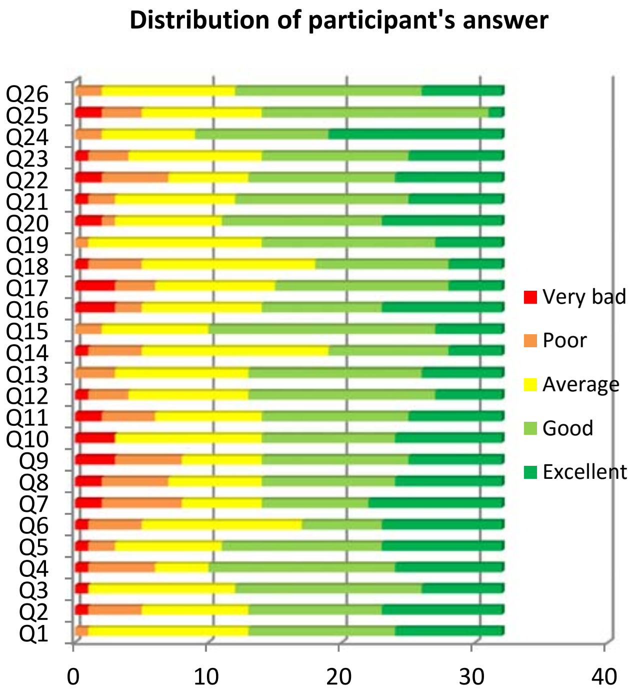


Figure 12. Distribution of participants' answers to PSSUQ.

Subscale	Mean (μ)	Median (M)	Standard deviation (σ)
SYSUSE	5.65	6.00	0.94
INFOQUAL	5.54	6.00	0.90
INTERQUAL	5.43	6.00	0.94
OVERALL SATISFACTION	5.59	6.00	0.72
USABILITY	5.73	6.00	0.90

Table 3. Scores aggregated in the CSUQ sub-scales.

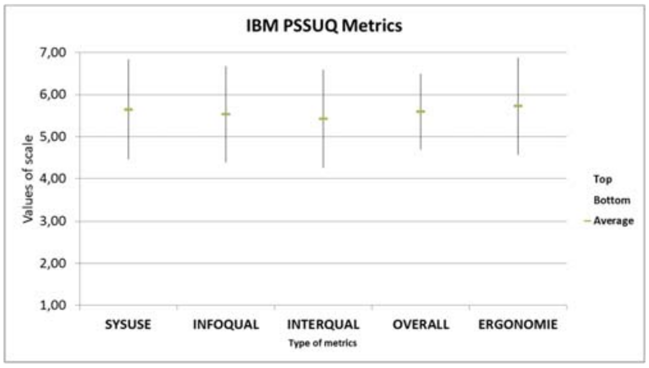


Figure 13. PSSUQ results in terms of metrics.

More specifically, a very good score related to user guidance may suggest that the material supporting guidance (e.g., where am I?, where do I come from?, where can I go?) is effective enough for participants to find the information relevant to design options, in particular through the generation preview. Information displayed in the software seems to be adequately conveyed. Some shortcomings are found however in the quality and the appreciation of the interface of UsiMAD itself, which is partially explained by the limited help capabilities delivered by UsiMAD.

On the contrary, some questions raised some concerns about their related aspect due to some 'very bad' answers. "Q9. The system gives error messages that clearly tell me how to fix problems" (μ=5.41, M=6, σ=1.34) and "Q10. Whenever I make a mistake using the system, I recover easily and quickly" (μ=5.50, M=6, σ=1.48) together reveal that the error management is insufficient, although the significant standard deviation suggest that some discrepancy across participants exists. Participants were able to backtrack the process by going backward and forward through the decision tree, but that was not the problem: it was more an error of intention than an error of execution. The user interface generation preview prevents participants from generating a MUI that does not correspond to their intention. Since values for design options are suggested by UsiMAD based on incorporated usability knowledge, some participants expressed their wish to depart from this facility by controlling the process entirely themselves. In general, mixed-initiative control is preferred over explicit control, but apparently not in our case.

"Q16. The interface of this system is pleasant" (μ=5.56, M=6, σ=1.32) and "Q17. I like using the interface of this system" (μ=5.34, M=6, σ=1.21) suggest that, although participants appreciated the whole process of using generative patterns, they did not appreciate how the UsiMAD user interface let them obtain the right application of the generative pattern. Further analysis should be conducted to determine a better presentation of the process.

Results from additional questions. Table 3 reports on the aggregated scores from the various sub-scales including the addition questions. All of them benefit from a value higher than 5, which is generally considered acceptable for a 7-point Likert scale in this case. Standard deviations of sub-scales are below the threshold of 1, thus suggesting that participants have a concentrated opinion on those various sub-scales. This is one of the most important observations because it shows that our implementation of the MAD generative pattern does not result in losing the user in complex hierarchies, thanks to the "expand" mechanism of the decision tree, which keeps the "path" of the participant's current location clearly visible at any time. One design option is questioned at a time and only the possible values at this level are provided.

6. CONCLUSION

The goal of this work is to provide a coherent framework based on generative design pattern principles and practices to interpret them and to contribute addressing the problem of multiple user interfaces implementation. The MAD framework was introduced for this purpose so that designers could express progressively a user and a device/platform model and a domain model as starting point, while the task model is pre-defined. Transitions from task and domain to abstract user interface, then from abstract user interface to concrete and to final user interface were defined and exemplified on the master-and-details pattern. We built UsiMad tool based on MAD framework that offers the observation of possible design options of the master-and-details pattern on different devices at different abstraction levels. The case study focuses on heterogeneous context and user interface design in an attempt to validate the outcomes. We show, for example, how at the bottom level of the Cameleon Framework, user can generate a UsiXML file, which is a XML-compliant document, to support the MAD pattern implementation.

The major contribution of this MAD framework is to offer some flexibility in designing multiple user interfaces for the same service, while ensuring a uniform user experience. As opposed to an approach where each device/platform holds a particular generation process where particular interaction techniques are optimized for the device/platform, UsiMAD privileges a uniform user experience by generating the code for multiple user interfaces based on a generative pattern that is consistent from one context of use to another.

We conducted a survey to evaluate the tool UsiMAD. As result, it was shown that the goal of the tool is understood by all participants. They understand the goal to guide developers in design pattern, the evaluation also reveals that the quality of information and appreciation of interface of the tool are limited. We assume that there are different notions are too abstract for users and pictures may be too small. In general, participants were satisfied more by the process of relying on generative patterns than the user interface of the UsiMAD supporting software itself. Participants appreciated the general guidance, but not the presentation of the decision tree offered by UsiMAD and regretted the lack of error management. There should be a difference between the need for having explanation on the decision suggested by UsiMAD (which is appreciated) from the way this message is conveyed through a decision tree (which is not so much appreciated). Navigation through the decision tree is straightforward since a node-to-node progression or backtracking is possible, but it causes visualization and scrolling issues.

Future works include expanding existing patterns to obtain a similar decision tree for other cases by gathering, compiling, structuring, and organizing various forms of effective usability knowledge, such as usability guidelines, design rules, style guides, and standards. These different forms of usability knowledge, which are potentially inconsistent, induce some integration problems into one single point of reference.

7. REFERENCES

[1] Alexander, C. 1977. *A pattern language: towns, buildings, construction*. Oxford University Press.

[2] Android Developers. *UI framework changes in Android 1.5*. Accessible at http://jayxie.com/mirrors/android-sdk/resources/articles/ui-1.5.html

[3] Aquino, N., Vanderdonckt, J., Condori-Fernández, N., Dieste, O., and Pastor, O. 2010. Usability evaluation of multi-device/device user interfaces generated by model-driven engineering. In *Proc. of 4th ACM Int. Symposium on Empirical Software Engineering and Measurement* (Bolzano, Sept. 16-17, 2010). ESEM'2010. ACM Press, Article #30.

[4] Balsamiq Mockup support. Accessible at http://balsamiq.com (January 2015).

[5] Barnett, S., Vasa, R., and Grundy, J. 2015. Bootstrapping mobile app development. In *Proc. of IEEE/ACM 37th Int. Conf. on Software Engineering* (Florence, May 16-24, 2015). ICSE'2015. IEEE Society Press, Piscataway, NJ, 657–660.

[6] Basili, V.R. and Rombach, H. D. 1998. The Tame Project: Towards Improvement-Oriented Software Environments. *IEEE Transactions on Software Engineering* 14, 6 (June 1988). 758–773.

[7] Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J. A unifying reference framework for multi-target user interfaces. *Interacting with computers* 15, 3 (Nov. 2003), 289–308.

[8] Coad, P., North, D., and Mayfield, M. 1997. *Strategies and Patterns Handbook: Hypertext Edition*. Int., Inc. 1997.

[9] Engel, J., Herdin, C., and Märtin, C. 2012. Exploiting HCI pattern collections for user interface generation. In *Proc. of 4th Int. Conf. on Pervasive Patterns and Applications* (Nice, July 22-27, 2012). PATTERNS'2012. Int. Acad., Research, and Industry Assoc., Wilmington, 36–44.

[10] Forbrig, P. and Wolff, A. 2010. Different kinds of pattern support for interactive systems. In *Proc. of the 1st Int. Workshop on Pattern-Driven Engineering of Interactive Computing Systems* (Berlin, June 20, 2010). PEICS'2010. ACM Press, New York, NY, 36–39.

[11] Francese, R., Risi, M., and Tortora, G. 2015. Management, Sharing and Reuse of Service-Based Mobile Applications. In *Proc. of 2nd ACM Int. Conf. on Mobile Software Engineering and Systems* (Florence, May 15-16, 2015). MobileSoft'2015. IEEE Computer Society Washington, DC, 105–108.

[12] Genaro Motti, V. and Vanderdonckt, J. 2013. A computational framework for context-aware adaptation of user interfaces. In *Proc. of IEEE 7th Int. Conf. on Research Challenges in Information Systems* (Paris, May 29-31, 2013). RCIS'-2013. IEEE Society Press, Piscataway, NJ, 1–12.

[13] Genaro Motti, V., Raggett, D., Van Cauwelaert, S., and Vanderdonckt, J. 2013. Simplifying the Development of Cross-Platform Web User Interfaces by Collaborative Model-based Design. In *Proc. of ACM Conf. on Design of Communication* (Greenville, Sept. 30th-October 1st, 2013) SIGDOC'2013, ACM Press, New York, 2013, pp. 55-64.

[14] Henninger, S. 2001. An organizational learning method for applying usability guidelines and patterns. In Proc. of *8th IFIP Int. Conf. on Engineering for Human-Computer Interaction* (Toronto, May 11-13, 2001). EHCI'2001. Lecture Notes in Comp. Sci., Vol. 2254. Springer, Berlin, 141–156.

[15] Kellen, V. 2003. *Business Performance Measurement. At the Crossroads of Strategy, Decision-Making, Learning and Information Visualization*. DePaul Univ., Chicago. Accessible at http://www.kellen.net/bpm.htm.

[16] Koskimies, O., Wikman, J., Mikola, T., and Taivalsaari, A. 2014. EDB: A Multi-Master Database for Liquid Multi-Device Software. In *Proc. of 2nd ACM Int. Conf. on Mobile Software Engineering and Systems* (Florence, May 15-16, 2015). MobileSoft'2015. IEEE Press, Piscataway, 125–128.

[17] Levin, M. 2014. *Designing Multi-Device Experiences.* O'Reilly Media.

[18] Lewis, J.R. 1995. IBM Computer Usability Satisfaction Questionnaires: Psychometric Evaluation and Instructions for Use. *Int. Journal of Human-Computer Interaction* 7, 1 (July 1995), 57-78.

[19] Likert, R. 1932. A technique for the measurement of attitudes. *Archives of psychology* 22, 140 (June 1932), 5–55.

[20] Mac Developer Library. *Cocoa Bindings Programming Topics*, Creating a Master-Detail Interface. Accessible at https://developer.apple.com/library/mac/#documentation/Cocoa/Conceptual/CocoaBindings/Tasks/masterdetail.html (2012).

[21] Marcotte, E. 2011. *Responsive Web Design*. A Book Apart.

[22] Meixner, G. and Calvary, G. 2014. *Introduction to Model-based User Interface*. W3C Working Group Note. World Wide Web Consortium, Geneva, 7 January 2014. Accessible at https://www.w3.org/TR/mbui-intro/.

[23] Nebeling, M., Mintsi, T., Husmann, M., and Norrie, M. 2014. Interactive development of cross-device user interfaces. In *Proc. of the ACM Conf. on Human Factors in Computing Systems* (Toronto, April 26-May 1, 2014). CHI'2014. ACM Press, New York, NY, 2793–2802.

[24] Nguyen, T.-D. and Vanderdonckt, J., "User interface master detail pattern on android," in *Proc. of the 4th ACM SIGCHI symposium on Engineering interactive computing systems*. ACM, 2012, pp. 299–304.

[25] Nilsson, E.G. 2009. Design patterns for user interface for mobile applications. *Advances in engineering software* 40, 12 (2009), 1318–1328.

[26] Pastor, O. and Molina, J.-C. 2007. *Model-Driven Architecture in Practice*. Springer, Berlin.

[27] Seffah, A. 2015. *Patterns of HCI design and HCI design of patterns - bridging HCI design and model-driven software engineering*. Springer, Berlin, 123–153.

[28] Tidwell, J. 2010. *Designing interfaces, patterns for effective interaction design*. O'Reilly Media Inc.

[29] van Duyne, J., Landay, J., and Hong, J. 2006. *The design of sites, patterns for creating winning websites*. Prentice Hall.

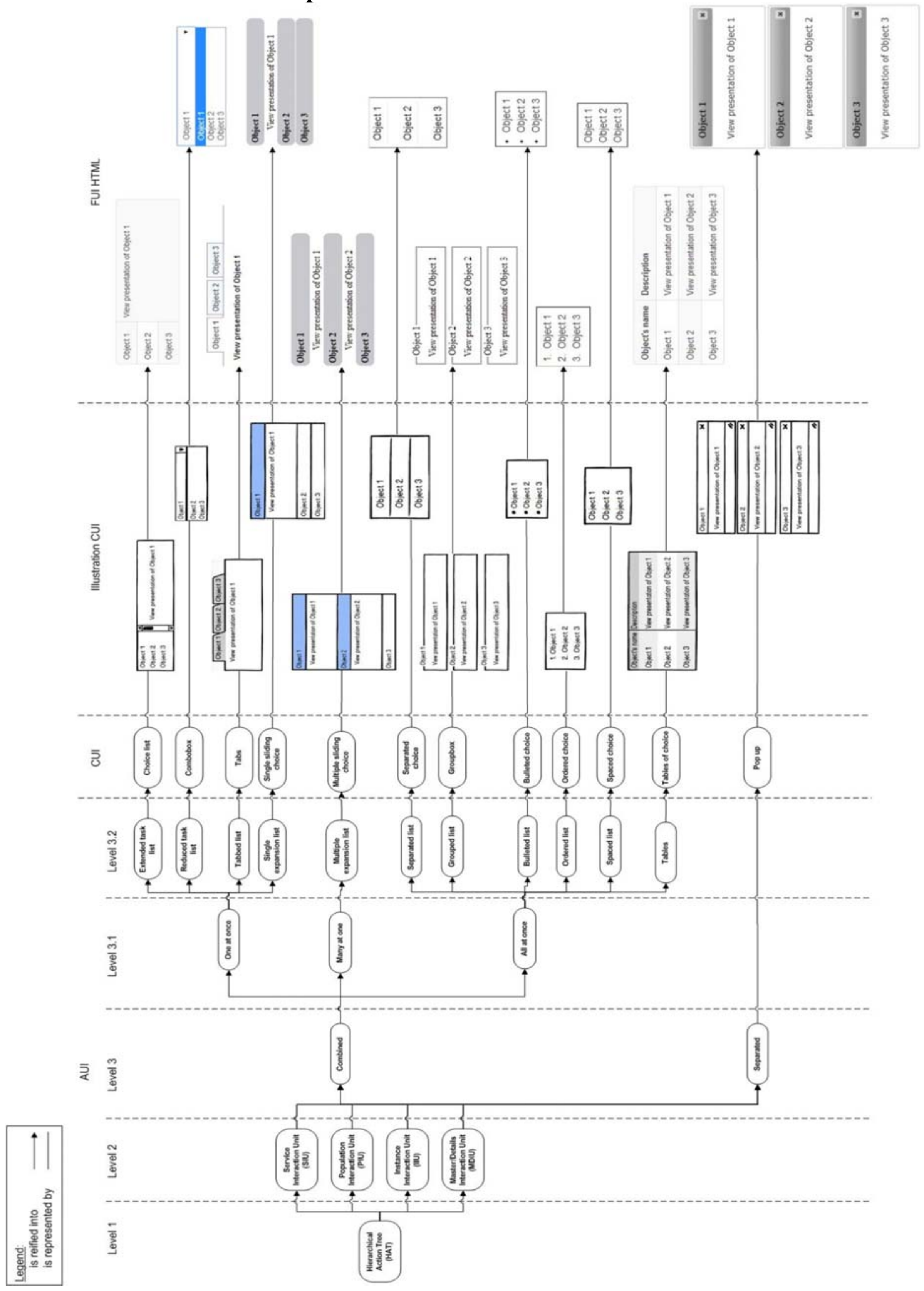
[30] van Welie, M., van der Veer, G.C., and Eliens, A. Patterns as Tools for User Interface Design. In *Proc. of 1st Int. Workshop on Tools for Working with Guidelines* (Biarritz, June 2000). TFWWG'2000. Springer, Berlin, 313–324.

[31] Vilkomir, S., Marszalkowski, K., Perry, Ch., and Mahendrakar, S. 2015. Effectiveness of Multi-device Testing Mobile Applications. In *Proc. of 2nd ACM Int. Conf. on Mobile Software Engineering and Systems* (Florence, May 15-16, 2015). MobileSoft'2015. IEEE Computer Society Washington, DC, 44–47.

[32] Voutilainen, J.-P., Salonen, J., and Mikkonen, T. On the Design of a Responsive User Interface for a Multi-Device Web Service. In *Proc. of 2nd ACM Int. Conf. on Mobile Software Engineering and Systems* (Florence, May 15-16, 2015). MobileSoft'2015. IEEE Computer Society Washington, 60–63.

[33] Wendler, S., Ammon, D., Philippow, I., and Streitferdt, D. 2013. A factor model capturing requirements for generative user interface patterns. In *Proc. of 5th Int. Conf. on Pervasive Patterns and Applications* (Valencia, May 27-June 1, 2013). PATTERNS'2013. Int. Acad., Research, and Industry Assoc., Wilmington, 34–43.

8. APPENDIX: Decision Tree Implemented in UsiMAD


Legend:
is reified into
is represented by
Level 1
Level 2
AUI
Level 3
Level 3.1
Level 3.2
CUI
Illustration CUI
FUI HTML
Hierarchical Action Tree (HAT)
Service Interaction Unit (SIU)
Population Interaction Unit (PIU)
Instance Interaction Unit (IIU)
Master/Details Interaction Unit (MDIU)
Combined
Separated
One at once
Many at one
All at once
Extended task list
Reduced task list
Tabbed list
Single expansion list
Multiple expansion list
Separated list
Grouped list
Bulleted list
Ordered list
Spaced list
Tables
Choice list
Combobox
Tabs
Single sliding choice
Multiple sliding choice
Separated choice
Groupbox
Bulleted choice
Ordered choice
Spaced choice
Tables of choice
Pop up
Object 1
Object 2
Object 3
View presentation of Object 1
View presentation of Object 2
View presentation of Object 3
Object's name
Description