UI Design Pattern Selection Process for the Development of Adaptive Apps

Amani Braham
University of Sousse, ISITCOM
4011 Hammam Sousse, Tunisia
e-mail: amanibraham@gmail.com

Maha Khemaja
University of Sousse
4000 Sousse, Tunisia
e-mail: khemajamaha@gmail.com

Félix Buendía
Universitat Politècnica de Valencia
46022 Valencia, Spain
e-mail: fbuendia@disca.upv.es

Faiez Gargouri
University of Sfax
3029 Sfax, Tunisia
e-mail: faiez.gargouri@isims.usf.tn

***Abstract*—In User Interface (UI) development, UI design patterns constitute a crucial solution that helps to resolve design problems by reusing design knowledge. The diversity of patterns would require deep developer experience to select relevant patterns and would make it difficult to apply the right patterns. This paper proposes an ontology of UI design patterns that enables a potential UI design pattern selection process. We focus particularly on the capability of the Adaptive User Interface Design Pattern (AUIDP) framework in selecting relevant UI design patterns using both ontological and ranking reasoning. This is demonstrated through a service-oriented tool that recommends appropriate patterns. This tool is evaluated with regard to three main factors, including the tool's usefulness and practicality, the developed interface quality and the developer productivity. Results show that the tool enhances developer's accuracy in terms of selecting relevant patterns and hastens the UI development process.**

Keywords-Adaptive User Interface; Interface specification and design; UI design patterns; Ontology model.

I. INTRODUCTION

Currently, smartphones and mobile technologies are in the process of an ever-increasing development. The extensive use of mobile devices resulted in a notable increase in the application development industry. This makes the mobile application industry a multi-billion dollar industry [1]. With the increase in the number of mobile applications (a.k.a. apps), developers face a major challenge related to UI development. The statistics presented by Myers et al. [2] reported that the time required for developing user interfaces reaches 50% of the time needed for software development, and their corresponding source code includes 48% of the whole code. These user interfaces are intended to be used by various users with different profiles and needs, and also using different types of devices. A study conducted in [3] showed that 15% of the world's population has some kind of disability, which could be physical, cognitive, or sensory. The great variety of disabilities that users may be affected by has led to the emergence of adaptive interactive systems [4]. Hence, these systems open up new challenges, as users need adaptive user interfaces that fit their corresponding disabilities and requirements. Therefore, this kind of interface is becoming one of the most dominant part of adaptive systems. However, its development is not a trivial task; it presents a high complexity and takes a long time in such a way that developers often cannot fully cover disabled user's needs and preferences. Moreover, developing adaptive user interfaces requires a multidisciplinary team with a deep experience in using design knowledge, resolving design problems, as well as choosing the relevant design solution. Within this context, UI design patterns are introduced to support the design of adaptive user interfaces [5], since they attempt to educate designers to build user interfaces [6]. While hundreds of UI design pattern catalogues have been developed and published [7], they tend to be overlooked in practice. The major hurdle in considering these catalogues is how developers can recognize the relevant patterns for solving a specific design problem. This is due to the lack of tools for selecting existing UI design patterns. This might lead to applicability issues that create difficulties for developers to properly select and apply UI design patterns, and makes the design and development of adaptive user interfaces a time consuming and tedious task. Therefore, it becomes mandatory to find an intelligent way to handle, select and use relevant design patterns, to increase the reusability of design knowledge, to decrease the time and complexity of the design and development process and, finally, to improve the quality of adaptive user interfaces for users with disabilities.

To tackle the above mentioned challenges, the remainder of this paper presents the Modular UI DEsign Pattern (MIDEP) ontology that enables a potential UI design pattern selection process. This ontology is created using a specific method and augmented with a set of inference rules that provide intelligent support for developers to integrate relevant UI design patterns while developing user interfaces. The selection process is demonstrated through the AUIDP framework, which allows semantic reasoning over the proposed ontology in order to deliver UI design patterns that contribute to the process of developing adaptive mobile applications for users with disabilities.

The rest of this paper is organized as follows. Section II reports related works that deal with design patterns modeling methods and UI design patterns in software development. Section III presents an overview of the AUIDP framework. In Section IV, we introduce the UI design pattern selection process. Section V presents the design pattern selection process as a service-oriented tool. Section VI presents an evaluation of the developed tool considering three main factors. Finally, the last section outlines the conclusion and opens up further research orientations.

II. RELATED WORK

This section goes through existing literature in order to cover works related to design patterns modeling methods and UI design patterns in software development.

A. *Design patterns modeling methods*

The cornerstone of design pattern concept was laid by Christopher Alexander [8], in late 1970s, to deal with problems occurring in building architecture and it was initially defined as "a three-part rule, which expresses a relation between a certain context, a problem, and a solution". Such concept has been used in the Human Computer Interaction (HCI) field and exploited as an approach to design and evaluate interfaces [6]. Within this context, several works proposed their own collections of design patterns, offering solutions for specific design problems. The pattern collection presented in [9] is considered as one of the largest libraries that covers different kinds of applications including Web and mobile. Likewise, the Welie's catalogue [10] includes 131 patterns for interaction design and particularly for Web design. Besides, Neil's collection [7] comes with patterns for mobile applications. Furthermore, Mushthofa et al. [11] introduced a set of design patterns for designing websites. Despite the large numbers of catalogues, patterns are usually expressed in a traditional text-based representation with different and inconsistent pattern attributes. To tackle the heterogeneity issues, some standardization methods have been introduced. In this sense, pattern languages have been introduced [12]. Nevertheless, these representations are not a satisfactory solution since applying patterns requires a deep developer's comprehension in the context of use of each pattern. This barrier makes accessing patterns more difficult for developers. A machine-comprehensive representation is thus required. In [13], the authors introduce usability patterns models using ontologies. Furthermore, in [14] a formalization of Gang of Four's patterns (GoF) is modeled by means of ontologies. In [15], the authors reveal the formalization of Web design patterns based on ontologies. In [16], Kultsova et al. developed an ontological model of UI and interface pattern. However, all these works concern the representation of a set of patterns in a specific area, considering only its internal structure (e.g., patterns attributes, and their constraints).

B. *UI design patterns in software development*

The development of adaptive user interfaces has been investigated in various software development methods [17]. Nevertheless, there is a lack of effective design knowledge reuse. The capability of UI design patterns has been exploited in software development, since they allow developers to reuse design knowledge [18]. In this context, both works [19] and [20] are based on UI design patterns for mobile development and application development, respectively. Similarly, Coleti et al. [21] exploited the use of mobile design patterns to support the development of interfaces. However, in the aforementioned works, patterns are identified and analyzed manually by developers, which constitutes a tedious task. So, developers may face ambiguity in selecting the right patterns. Tools and techniques are then needed to retrieve relevant patterns and apply them to support the UI development process.

C. *Discussion*

In line with this literature review, the proposed work undertakes three main purposes: i) the specification of design patterns, ii) the selection of patterns according to specific design problems, and, iii) their applicability in UI development process. To this end, we provide a consistent and formal specification of UI design patterns by using ontologies. Furthermore, we present a framework that allows semantic reasoning to retrieve patterns and provides mechanisms to integrate patterns in the development of adaptive user interfaces for users with disabilities.

III. OVERVIEW OF THE AUIDP FRAMEWORK

The present framework contributes to the development of adaptive mobile applications for users with disabilities following a hybrid approach by combining model-based user interface development methods with pattern-based methods. The foundation of the proposed framework relies on the idea that the user interface can be fully modeled by a set of model fragments which is able to address a specific instance of UI design pattern. Therefore, within the AUIDP framework, UI design patterns constitute the basics for generating the final user interface. The overall overview of the AUIDP framework is depicted in Figure 1. It consists of four phases, including: UI design pattern selection, pattern instantiation, pattern integration, and, finally, user interface generation.

Furthermore, the AUIDP framework provides an environment for multidisciplinary teams to design and implement adaptive user interfaces in a consistent way by addressing particularly the following main aspects:

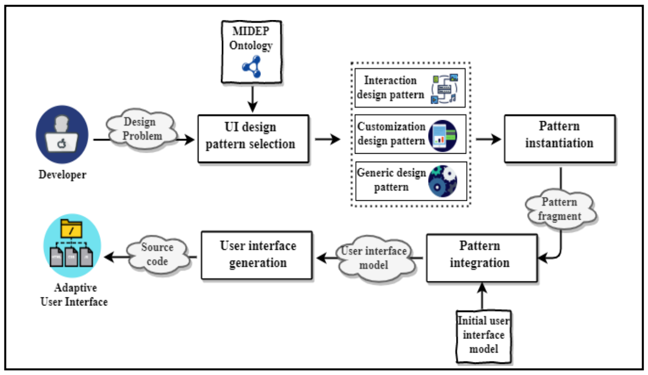


Figure 1. Framework overview.

- Open and accessible: The proposed framework puts UI design patterns at the fingertips of software developers/designers so they could be used in designing and developing the user interface.
- Modular: The user interface development process within the AUIDP framework is achieved by composing different UI design patterns.
- Code reuse: The developer has full access to the source code of the adaptive UI to be developed. Each UI design pattern that composes the interface is delivered with the source code corresponding to its design solution. This aspect speeds up the development process, fosters reuse and thus reduces the code that has to be developed.

In this paper, we focus on the UI design pattern selection process. A detailed description of the component that deals with the selection process is outlined in the next section.

IV. Overview of UI design pattern selection component

The purpose of this component is to automate the selection process of UI design patterns for specific design problems within the AUIDP framework. Handling this process requires mechanisms and methods for searching, classifying and selecting UI design patterns that will be further used in future work for developing user interfaces. In this regard, we rely on a rich repository of UI design patterns. However, the large number of UI design pattern and the complex relationships among them is becoming the primary impediment for recognizing relevant patterns. In addition to a textual description, a formal representation of UI design patterns is therefore required as input of the design pattern selection component. In the subsections below, we introduce the MIDEP ontology and the methodology used for the construction of this ontology; then, we examine the architecture of the component that is adopted for selecting relevant design patterns.

A. *MIDEP ontology*

The MIDEP ontology comprises knowledge about UI design patterns, since the AUIDP framework aims to support the design of adaptive mobile applications. This ontology mainly represents the best practices of UI development for users with special needs and uses information of design patterns that are introduced in [22]. In order to build the MIDEP ontology, we adopted the Neon methodology [23] since it can help to re-engineer non-ontological resources into ontologies, reuse existing ontologies, and thus assure modularity that would lead to consider the multidisciplinary aspect. In this regard, we identify the following three scenarios, which are extracted from a set of nine scenarios provided by the Neon method for building the MIDEP ontology:

- Neon's scenario 1: From specification to implementation.
- Neon's scenario 2: Reusing and re-engineering non-ontological resources.
- Neon's scenario 4: Reusing and re-engineering ontological resources.

Figure 2 illustrates the main steps considered when building the MIDEP ontology using a combination of the three aforementioned scenarios. A detailed description of each phase is outlined below.

1) *Ontology requirement specification:* The purpose of this phase is to introduce the ontology scope and motivation. It articulates the necessity of steps from step 1 to step 6 and gives as a result a global glossory of terms.

a) Specification: The MIDEP ontology is proposed as a modeling solution to tackle recurring design problems related to user interfaces. Within this step, we have identified a set of informal Computency Questions (CQs) which are used to evaluate the effectiveness of the ontology [24]. Some CQ examples are: What are the elements that compose a design pattern? What solution design pattern will provide? What are the relationships among design patterns and user interfaces? Which kind of design problem information could lead to better decision making for selecting relevant design patterns? Which kind of information could help to distinguish patterns that contribute to the same design problem?

b) Non-ontological resource selection: Several design pattern collections and catalogues have been developed. Within this step, patterns that can be used to deal with Web and mobile applications are reviewed, including the Tidwell's library [11], the Welie's catalogue [12], and Nilsson's collection [25].

c) Non-ontological resource re-engineering: For the aforementioned catalogues, we identified the attributes adopted for structure design patterns.

d) Ontology selection: An ontology named ONTO [26] for modeling the user interface is selected within this step.

e) Ontology resource re-engineering: Some concepts, terms and attributes are extracted from the ontology selected in the previous step.

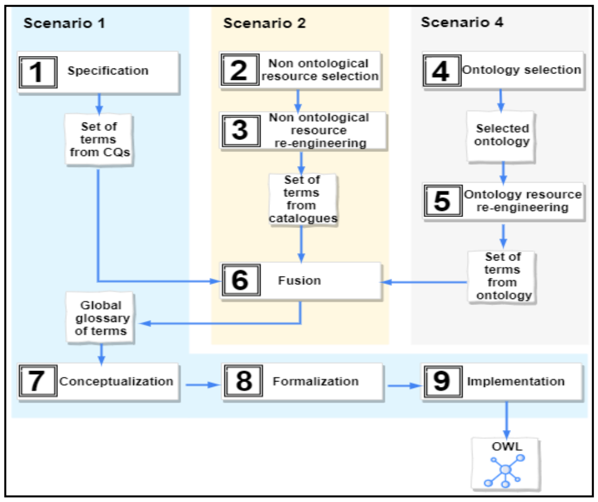


Figure 2. Process overview for building MIDEP ontology.

f) Fusion: This phase aims to blend terms of glossaries resulted from CQs, design pattern catalogues, and the selected ontology.

2) Conceptualization: It concerns mainly the definition of concepts and subconcepts, as illustrated in Figure 3.

3) Formalization: Once classes and subclasses are defined, a formal model is built. To this end we used Ontology Web Language version 2 (OWL2) as an ontology respresentation language.

4) Implementation: The concepts introduced previously are implemented using the Protégé editor tool.

B. Design pattern selection component architecture

This component incorporates two main modules, namely the reasoning engine and the ranking calculation engine. These modules interact among them to deal with the pattern selection process, as illustrated in Figure 4.

The reasoning engine component takes as input design problems that address mainly issues related to user characteristics, as well as interaction design issues [27]. User characteristic issues concern information about users, by whom the final interface is intended to be used, including user's disability, interest, goal, task, and need. Interaction design issues are information that comprise scattered data, bad contrast of colors, and useless interface elements. Once these issues are acquired from developers, the reasoning engine provides real time reasoning. It uses the MIDEP ontology in combination with a set of rules to decide on the UI design patterns that should be retrieved.

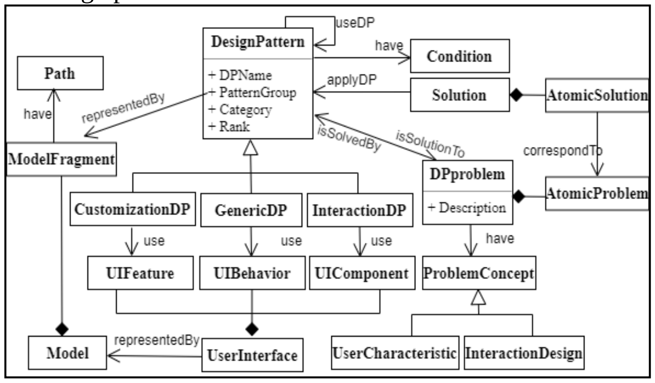


Figure 3. MIDEP Ontology model.

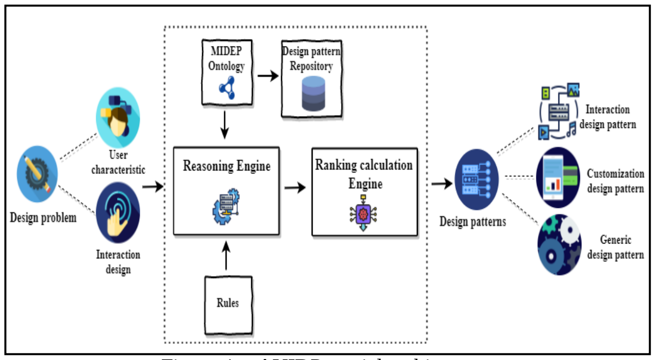


Figure 4. AUIDP partial architecture.

The ranking calculation engine is in charge of refining the set of design patterns resulted from the reasoning engine. It computes the similarity between the input design problem and the problem definition corresponding to design patterns retrieved from the reasoning engine. To this end, the ranking calculation engine applies the Cosine Similarity (CS) measure [28], since it allows computing the similarity of text documents. The CS values of each design pattern are calculated using (1), where patterns' problem and design problems are defined by a vector of terms and a frequency vector. For example, in (1), A and B are the frequency vectors of patterns' problem and design problems, respectively. This engine uses the obtained CS values to rank patterns, and generates the relevant UI design patterns that have the highest similarity scores.

$$CS(A,B) = \frac{\sum_{i=1}^{N} (A_i \; B_i)}{\sqrt{\sum_{i=1}^{N} (A_i)^2} * \sqrt{\sum_{i=1}^{N} (B_i)^2}} \tag{1}$$

V. DESIGN PATTERN SELECTION AS A SERVICE

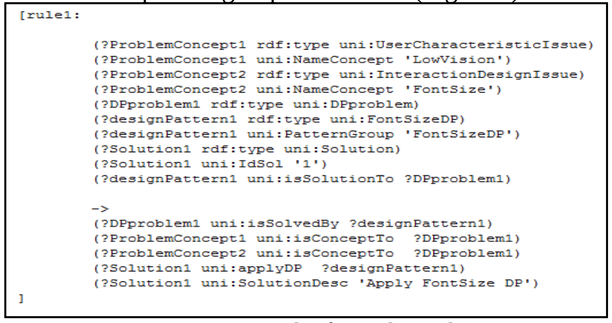
A. Implementation features

In order to implement the selection process, we developed a service-oriented tool including reasoning and ranking calculation services. It generates recommendations of design patterns according to specific design problems using a set of REpresentational State Transfer (REST) Web services. To this end, we used the generic reasoner that is considered as one of the inference engines supported by Jena and serves as the basis for OWL and Resource Description Framework Schema (RDFS) reasoners. It mainly exploits a rule-based engine for reasoning over the proposed ontology as well as for processing SPARQL queries.

B. Experiments and results

The procedure of design patterns selection phase within the developed tool can be introduced by the following experiment: A design problem "DP-1", that includes "LowVision" as users' characteristic issue and "FontSize" as interaction design issue, is considered in this experiment.

In the first step, the reasoning mechanism enables to obtain the design patterns' group, according to "DP-1". In this case, the reasoning engine triggers "rule 1" depicted in Figure 5. As a result, a set of design patterns corresponding to the selected pattern group is retrieved (Figure 6).

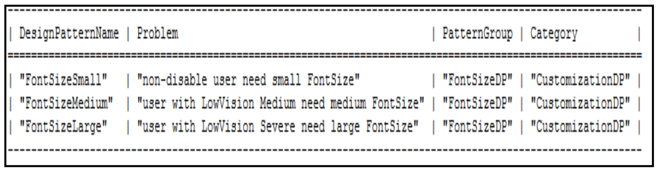


```
[rule1:

        (?ProblemConcept1 rdf:type uni:UserCharacteristicIssue)
        (?ProblemConcept1 uni:NameConcept 'LowVision')
        (?ProblemConcept2 rdf:type uni:InteractionDesignIssue)
        (?ProblemConcept2 uni:NameConcept 'FontSize')
        (?DPproblem1 rdf:type uni:DPproblem)
        (?designPattern1 rdf:type uni:FontSizeDP)
        (?designPattern1 uni:PatternGroup 'FontSizeDP')
        (?Solution1 rdf:type uni:Solution)
        (?Solution1 uni:IdSol '1')
        (?designPattern1 uni:isSolutionTo ?DPproblem1)

        ->
        (?DPproblem1 uni:isSolvedBy ?designPattern1)
        (?ProblemConcept1 uni:isConceptTo  ?DPproblem1)
        (?ProblemConcept2 uni:isConceptTo  ?DPproblem1)
        (?Solution1 uni:applyDP  ?designPattern1)
        (?Solution1 uni:SolutionDesc 'Apply FontSize DP')
]
```

Figure 5. Example of DP rules: Rule1.



```
------------------------------------------------------------------------------------------------------
| DesignPatternName | Problem                                           | PatternGroup | Category        |
======================================================================================================
| "FontSizeSmall"   | "non-disable user need small FontSize"             | "FontSizeDP" | "CustomizationDP" |
| "FontSizeMedium"  | "user with LowVision Medium need medium FontSize"  | "FontSizeDP" | "CustomizationDP" |
| "FontSizeLarge"   | "user with LowVision Severe need large FontSize"   | "FontSizeDP" | "CustomizationDP" |
------------------------------------------------------------------------------------------------------
```

Figure 6. Design patterns instances.

In the second step, a set of design pattern' instances generated by the reasoning engine will be refined in order to retrieve the most relevant design patterns. To this end, the ranking engine calculates the CS between the design patterns 'problem and DP-1. Table I presents the resulting CS values.

TABLE I. CS VALUES FOR DP-1

Value	**Design Pattern**		
	FontSizeSmall	***FontSizeMedium***	***FontSizeLarge***
CS	0.316	0.534	0.534

Finally, the ranking engine returns the patterns with the highest CS score. In this experiment, "FontSizeMedium" and "FontSizeLarge" are the relevant patterns that are recommended using our tool to resolve DP-1.

VI. EVALUATION

The developed service oriented tool for selecting design patterns was evaluated in terms of being effectively usable by the developer, considering the following factors: the usefulness and practicality of the tool, the application's interface quality, and developer productivity. These factors constitute the main requirements for the design pattern selection process. To assess these factors, three main research questions, were addressed as follows:

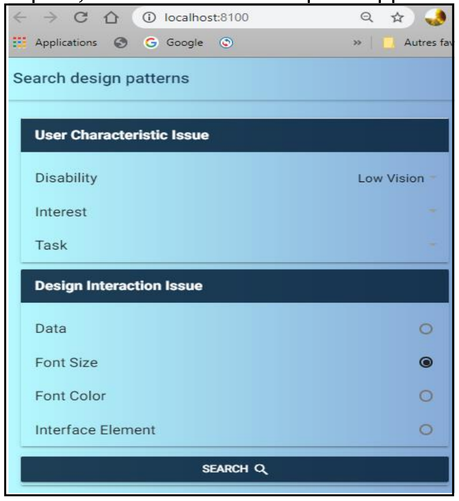
- RQ1: How can the tool assess the practicality for design patterns recommendation?
- RQ2: How well can the developed tool enhance the developer's accuracy in using design patterns?
- RQ3: How can the tool hasten the UI development process?

A. *Tool validation (RQ1)*

The usefulness and practicality of the proposed tool has been verified by the development of a hybrid application named Design Pattern Retrieve Application (DPRA) using Ionic [29]. This application includes a main menu for selecting the design problem, as illustrated in Figure 7. It further covers functionalities to allow a multidisciplinary team to view and extract relevant design patterns, as presented in Figure 8 and Figure 9, in order to resolve design problems.

B. *Developer based evaluation (RQ2, RQ3)*

1) *Experimental setting:* An experiment was designed in which two groups of software developers were invited to develop a location-based application that is able to track the user's current location and locate different points of interest. Each group consisted of four developers having University degrees in Computer Science and experience in creating hybrid applications using the Ionic framework. The first group, "Group-1", was asked to develop the application



Search design patterns

User Characteristic Issue

Disability: Low Vision

Interest

Task

Design Interaction Issue

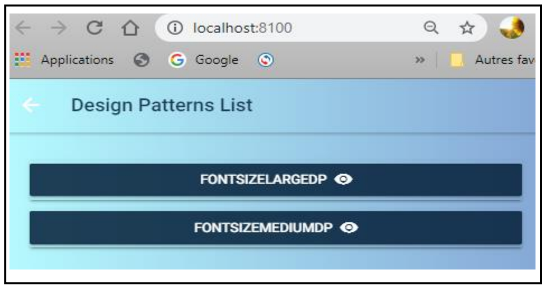
Data ○

Font Size ●

Font Color ○

Interface Element ○

Figure 7. DPRA main menu.

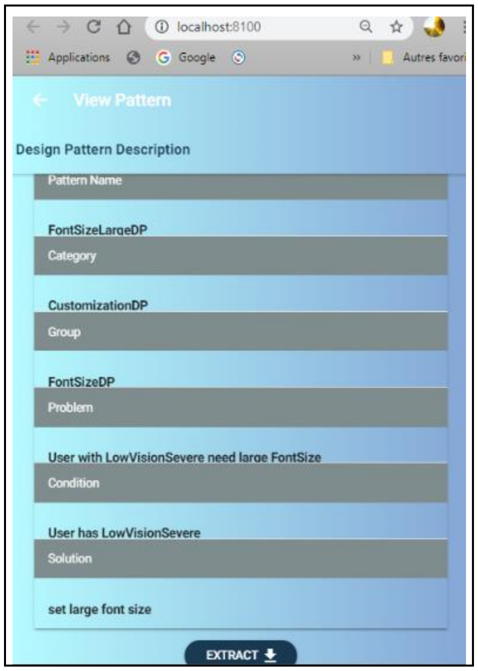


Design Patterns List

FONTSIZELARGEDP

FONTSIZEMEDIUMDP

Figure 8. Design patterns list.



View Pattern

Design Pattern Description

Pattern Name: FontSizeLargeDP

Category: CustomizationDP

Group: FontSizeDP

Problem: User with LowVisionSevere need large FontSize

Condition: User has LowVisionSevere

Solution: set large font size

Figure 9. Design pattern Description.

without any tool. The second group, "Group-2", was provided the developed tool to support their application development. We conducted this study because we wanted to track the interface quality and developer productivity factors. The influence on the interface quality and developer productivity were inspected by measuring the accuracy of design patterns and by recording UI development time, respectively. The accuracy is calculated using (2) and scaled from 0 (0% accurate) to 1 (100% accurate), where the error rate is the percentage of failed developed interfaces. In (3), it is assumed that failed interfaces are interfaces that do not consider appropriate design patterns.

$$Accuracy = 1 - ErrorRate \quad (2)$$

$$ErrorRate = \frac{Number\ of\ failed\ interfaces}{Number\ of\ interfaces} \quad (3)$$

2) *Results:* The first step consisted in calculating the accuracy. In this case, the accuracy was about 33% for "Group-1" and about 88% for the second group, which is greater than the first accuracy value. These results outline that the set of design patterns recommended from the provided tool indeed enhances the accuracy of selecting relevant design patterns used in the development of UIs. Hence, the exploitation of the selected patterns makes the location-based application developped by the second group better than the application of the first one in terms of considering a good ergonomic design. The second step was to measure the amount of time for each group to fulfill the application development. Results show that the development time varied among the two groups: for "Group-1", the development took 9 days (12h/day, about 108 hours) while "Group-2", whose implementation method is based on the proposed tool, has taken only 5 days (12h/day, about 72 hours). The development time is dramatically reduced in the second group. This is due to the fact that the tool permitted "Group-2" to quickly identify relevant design patterns and extract their corresponding code and reuse it in the application's implementation instead of reinventing the whole application code. In general, these results indicate that the developed tool has a quite good impact on enhancing the interface quality, as well as on increasing developer productivity. Thus, the framework presented in this work allows a potential selection of design patterns. However, this framework has some limitations since the design pattern selection process is restricted to some UI design patterns. This lack can increase the development rework as well as the inability to adapt to changing disabled user's needs.

VII. Conclusion and Future Work

In this work, we have presented an ontology for UI design pattern specification. Subsequently, we have introduced the AUIDP framework's main components, which concern the UI design pattern selection phase, including the reasoning and the ranking calculation engines. Such phase is implemented using a service oriented tool and evaluated considering the tool's usefulness and practicality, the interface quality, and the developer productivity. The experimental results, obtained in this work, consolidate the efficiency of the developed tool in terms of enhancing developer's accuracy in selecting relevant patterns and increasing developer productivity. As part of future work, we intend to generate adaptive UIs by using design patterns. So, we will target our emphasis on covering phases that follow the selection phase within the AUIDP framework. To address the limitation of the proposed framework, we will further extend the MIDEP ontology to cover the heterogeneity of design patterns and we will work on enhancing the developed service oriented tool functionalities.

References

[1] Mobile application revenue generation. [Online]. Available from: https://www.abiresearch.com/press/tablets-will-generate-35-of-this-years-25-billion-/ [retrieved: December, 2019].

[2] B. A. Myers and M. B. Rosson, "Survey on user interface programming," In Proceedings of the SIGCHI conference on Human factors in computing systems, pp. 195-202, 1992.

[3] World Health Organization, World health statistics 2016: monitoring health for the SDGs sustainable development goals, World Health Organization, 2016.

[4] P. Brusilovski, A. Kobsa, and W. Nejdl, "The adaptive web: methods and strategies of web personalization," Springer Science & Business Media, 2007.

[5] M. Peissner, D. Häbe, D. Janssen, and T. Sellner, "MyUI: generating accessible user interfaces from multimodal design patterns," In Proceedings of the 4th ACM SIGCHI symposium on Engineering interactive computing systems, pp. 81-90, 2012.

[6] C. E. Wania, "Exploring Design Patterns as Evaluation Tools in Human Computer Interaction Education," MWAIS 2019 (9), 2019.

[7] T. Neil, "Mobile design pattern gallery: UI patterns for smartphone apps," "O'Reilly Media, Inc.", 2014.

[8] C. Alexander et al., "A pattern language," Gustavo Gili, pp. 311-314, 1977.

[9] J. Tidwell, "Designing interfaces: Patterns for effective interaction design," O'Reilly Media, Inc., 2010.

[10] M. van Welie, "Patterns in Interaction Design. [Online]. Available from : http://www.welie.com [retrieved: December, 2019].

[11] D. Mushthofa, M. K. Sabariah, and V. Effendy, "Modelling the user interface design pattern for designing Islamic e-commerce website using user centered design," In AIP Conference Proceedings, AIP Publishing LLC, vol. 1977, no. 1, pp. 020022, 2018.

[12] Y. Pan and E. Stolterman, "Pattern language and HCI: expectations and experiences," In CHI'13 Extended Abstracts on Human Factors in Computing Systems, pp. 1989-1998, 2013.

[13] S. Henninger and P. Ashokkumar, "An ontology-based infrastructure for usability design patterns," Proc. Semantic Web Enabled Software Engineering (SWESE), Galway, Ireland, pp. 41-55, 2005.

[14] H. Kampffmeyer and S. Zschaler, "Finding the pattern you need: The design pattern intent ontology," In International Conference on Model Driven Engineering Languages and Systems, Springer, Berlin, Heidelberg, pp. 211-225, 2007.

[15] S. Montero, P. Díaz, and I. Aedo, “Formalization of web design patterns using ontologies,” In International Atlantic Web Intelligence Conference, Springer, Berlin, Heidelberg, pp. 179-188, 2003.

[16] M. Kultsova, A. Potseluico, I. Zhukova, A., Skorikov, and R. Romanenko, “A two-phase method of user interface adaptation for people with special needs,” In Conference on Creativity in Intelligent Technologies and Data Science, Springer, Cham, pp. 805-821, 2017.

[17] I. Jaouadi, R. Ben Djemaa, and H. Ben Abdallah, “Interactive systems adaptation approaches: a survey,” In Proceedings of the 7th International Conference on Advances in Computer-Human Interactions ACHI, pp. 127-131, 2014.

[18] P. Cremonesi, M. Elahi, and F. Garzotto, “User interface patterns in recommendation-empowered content intensive multimedia applications,” Multimedia Tools and Applications, vol. 76, no. 4, pp. 5275-5309, 2017.

[19] T. Wetchakorn and N. Prompoon, “Method for mobile user interface design patterns creation for iOS platform,” In 2015 12th International Joint Conference on Computer Science and Software Engineering (JCSSE), IEEE, pp. 150-155, 2015.

[20] C. A. Cortes-Camarillo et al., “EduGene: a UIDP-based educational app generator for multiple devices and platforms,” International Journal of Human–Computer Interaction, vol. 35, no. 3, pp. 274-296, 2019.

[21] T. A. Coleti et al., “Design Patterns to Support Personal Data Transparency Visualization in Mobile Applications,” International Conference on Human-Computer Interaction, Springer, Cham, pp. 46-62, 2019.

[22] A. Braham, F. Buendía, M. Khemaja, and F. Gargouri, “Generation of Adaptive Mobile Applications Based on Design Patterns for User Interfaces,” In Multidisciplinary Digital Publishing Institute Proceedings, vol. 31, no. 1, pp. 19, 2019.

[23] M. C. Suárez-Figueroa, A. Gómez-Pérez, and M. Fernández-López, “The NeOn methodology for ontology engineering,” In Ontology engineering in a networked world, Springer, Berlin, Heidelberg, pp. 9-34, 2012.

[24] M. Grüninger and M. S. Fox, “Methodology for the design and evaluation of ontologies,” 1995.

[25] E. G. Nilsson, “Design patterns for user interface for mobile applications,” Advances in engineering software, vol. 40, no. 12, pp. 1318-1328, 2009.

[26] M. Ansarinia. User Interface Ontolog. [Online]. Available from: https://old.datahub.io/dataset/ui [retrieved: December, 2019].

[27] W. Iftikhar et al., “User Interface Design Issues In HCI,” International journal of computer science and network security, vol. 18, no. 8, pp. 153-157, 2018.

[28] A. Huang, “Similarity measures for text document clustering,” In Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand, vol. 4, pp. 9-56, 2008.

[29] Ionic Framework. [Online]. Available from : https://ionicframework.com/ [retrieved: January, 2020].