# Towards an Approach for Incorporating Usability Requirements into Context-Aware Environments

Dorra Zaibi, Meriem Riahi and Faouzi Moussa
*LIPAH Laboratory, Faculty of Science of Tunis, University of Tunis El Manar, Tunis, Tunisia*

Keywords: Ubiquitous Computing, Context-Aware Systems, Usability Rules, Inference System, Model-Driven Architecture.

Abstract: With the considerable advancement of technologies and the proliferation of mobile devices, evaluating the usability of software applications has become an emerging research area. Hence, improving the quality of software applications is crucial in context-aware environments. For that reason an increasing attention is drawn towards the development and the adoption of appropriate research proposals able to evaluate the mobile application usability. This article is a contribution to proposing a methodology for the development of context-aware systems based on usability requirements during the user interface design stage. In particular, this approach focuses on how to infer consistent context-based usability requirements and how to incorporate these requirements into a user interface development process. As a proof of the proposal concept, we have applied our methodology to an illustrative case study. More, experiments with end users have been carried out.

## 1 INTRODUCTION

Nowadays, ubiquitous computing is indispensable in the improvement of customized access to a wide range of mobile devices and services (Pathan and Reiff-Marganiec, 2009). In his/her ordinary activities and on a daily basis a user engages simultaneously with many smart devices. Thus, the concept of context of use plays a paramount role in the development of ubiquitous systems (Dey and Abowd, 2000). It involves information related to various aspects such as screens, locations, users, etc.
Actually, it is proved in literature studies that the context of mobile devices is highly dynamic when compared with the traditional desktop computers (Harrison et al., 2013). For that purpose, considerable challenges and attention have been raised with respect to usability inconsistencies of context-aware systems.

Accordingly, in order to provide better usability quality in ubiquitous environment, the evaluation process from the earlier stages of the user interface design process is crucial; mainly in the case of non-tolerant interactive systems failures in critical domains (Serral et al., 2010). User Interface (UI) has to show information in the best possible way in order to minimize error risks and erroneous manipulations. In this context, we believe we need a new generation of methodologies which allows context-aware systems developers to work while taking into consideration usability requirements right from the early UI design phase.

Actually, MDA (OMG MDA, 2003) is being largely explored. It has been proven that it is quite appropriate therefore becoming an essential paradigm for the software system design and development. In the last decade, the Human-Computer Interaction community (HCI) has highlighted the benefits of the MDA technology and accordingly is moving towards it (Oliviera et al., 2013). Generally, in such a process, attention is focalized in data and functional modelling neglecting usability aspects in context-aware environments. Hence, there is a need to require the MDA process extension for the purpose of supporting context-based usability as a first class entity in the user interface development process.

The present article aims to propose a model-driven approach for the integration of usability requirements into context-aware systems. The remainder of this paper is structured as follows. Section 2 describes the related works. Section 3 presents the proposed approach. Section 4 presents the case study. Section 5 describes the experiments of the adaptive system with end-users. Finally, the conclusion and further works are presented in Section 6.



Zaibi, D., Riahi, M. and Moussa, F.
Towards an Approach for Incorporating Usability Requirements into Context-Aware Environments.
DOI: 10.5220/0006702404630474
In *Proceedings of the 20th International Conference on Enterprise Information Systems (ICEIS 2018)*, pages 463-474
ISBN: 978-989-758-298-1

# 2 RELATED WORKS

In this section, we first present some approaches dealing with context-aware systems development. Second, we present some proposals that focus on usability evaluation. Finally, we present a third collection that brings these two topics together.

## 2.1 Context-Aware Approaches

Since the emergence of ubiquitous computing, great research efforts have been devoted to the subject of context-aware systems and several approaches have been proposed. We focus our attention on the studies that deal with context-awareness in Model-Driven environment. A notable example is of (Paterno et al., 2009) who suggest a universal model-based language for UI, called MARIA, to support the development of UI for interactive applications based service-oriented architectures in ubiquitous environments. The authors focus on exploiting the novel model-based language to provide useful support at both design and runtime. (Serral et al., 2010) propose a specification of context-aware pervasive systems through Model-Driven approach. PERVML, a domain specific modelling language, has been defined to describe the system functionality in a platform and technology independent way in order to model the pervasive system. (Oliveira et al. 2013) present a Model-Driven approach which takes into account the content personalization of the UI since earlier design stages.

As in the first group, these approaches, in spite of focusing on context-aware UI adaptation, neglect usability measures. They give support for the development of context-aware systems; yet, do not tackle the problem of preserving usability when adapting the UI.

## 2.2 Usability Evaluation Approaches

Numerous are the research studies that deal with usability in Model-Driven environment. For instance, a model-driven approach to evaluate the usability of multi-platform graphical UI is proposed by (Aquino et al., 2010). The usability has been quantified in terms of satisfaction, effectiveness and efficiency. The graphical UI is evaluated using small, standard and large size screens. (Gonzalez-Huerta et al., 2010) propose an architecture to support model-driven development process which is guided by quality attributes. Model transformations are specified and executed. The alternative transformations are selected considering the desired qualities of a particular target model. A method is proposed by (Panach et al., 2014) for evaluating internal usability. The authors define metrics for conceptual primitives. Based on conceptual models, the evaluation can be performed automatically and applied to any Model-Driven Development method.

Nevertheless, these proposals do not support usability evaluation with regard to changing context requirements that involves a considerable challenge to mobile devices' effectiveness. Currently, some research studies of usability-based context-aware approaches have been found. Next, we deal with these studies.

## 2.3 Usability-based Context-Aware Approaches

(Ormeno et al., 2013) present a method to facilitate the usability requirements of capture process at early stages. The method consists of defining a tree structure including interface design guidelines and usability guidelines. A question-answer format was used to enable the capture of requirements, through an interview with the end-user. Using model to model transformations, the usability requirements are transformed into a conceptual model of any existing Model-Driven Development method. The result of the interview is a set of designs that the system must satisfy. (Ben Ammar et al., 2015) propose a Model-Driven method for integrating usability guidelines into model transformation process. The proposed method aims to obtaining a UI which fulfils the desired usability attributes. Usability properties are associated with the possible alternative transformations, and parameterized transformations are executed. (Hentati et al., 2016) propose an MDE approach for optimizing usability in interactive systems generation process. Three main stages are fulfilled in their proposal: (1) generating all the possible concrete UI from a given abstract UI, (2) optimizing the usability by means of metrics to measure the user interface usability value considering a given context of use, (3) selecting the alternative model transformation in order to generate the optimal usability UI. A context based evaluation method for the assessment of the quality in use of mobile systems is proposed by (Ben Ayed et al., 2017). The authors implement the Evaluation Support System (ESS) in order to capture interaction data and contextual information when using mobile application. In addition to that, it allows the quantitative measurement of criteria, defined by the standard ISO/IEC 25010.

The study of the aforementioned proposals allows us to underline some limitations:

- Lack of dynamic selection of usability requirements in context-aware environment: All proposals are incapable to select dynamically the usability requirements with regard to the change of context of use. Only static adaptation is provided.
- Difficulty to select consistent usability guidelines: Applying a set of usability guidelines may have negative impact and provide conflicting influence in the whole usability of the UI. In most proposals, there doesn't exist any system able to select appropriate usability guidelines in order to perform conflict resolution. We can notice as an example of conflicting usability guidelines, that a usability guideline, which improves the user control, generally adds undo/cancel buttons to the UI. The application of this guideline may have negative impact in the information density attribute (e.g. limiting the amount of information in small screen size).
- Lack of usability guidelines in context-aware environments: no existing approaches have proposed novel usability guidelines for context-aware environments. All of them focus on existing context-based usability standards and recommendations issued from literature such as (ISO, 2010).
- Lack of empirical validation: Scientific methods need to be empirically validated in order to provide evidence about their effectiveness. Only a minority of the proposals has been empirically validated (Ben Ammar et al., 2015; Ben Ayed et al., 2017).

Although dealing with usability in context-aware systems is not novel, we can notice that this important research field is still in its early stage and many more research studies are recommended.

To cover this need, we propose a methodology that focuses on usability requirements during the adaptation of context-aware UI. The goals of our proposal are: 1) the selection of consistent usability requirements in context-aware environment is proposed, 2) novel generation of context-based usability rules are defined into a specific application domain, 3) the model-driven development (MDD) process is fulfilled during the context-aware UI modelling, and 4) experiments with end-users are carried out.

# 3 PROPOSED APPROACH

The main goal of this approach is to include usability guidelines in the context-aware process from the beginning of the UI modelling stage. We would like, while designing a UI, to set which consistent usability guidelines should be provided for each context situation.

Our approach is summarized in Figure 1. The system is developed by performing two stages; taking into consideration our knowledge of regular context information. In this sense, a specific system is held at runtime, updating context information constantly. During design time, we use this information when designing the UI.

To define our approach, two main stages are considered:

Stage 1: Selecting consistent usability rules. A set of usability guidelines is established to define consistent usability requirements considering current context of use.

Stage 2: In this stage, Model-driven development is adopted first, which uses the MDA approach and produces transformation models. We specify three MDA transformation levels namely: Computational Independent Model (CIM as CIM), Contextualized Platform Independent Model (CPIM as PIM) and Contextualized Platform Specific Model (CPSM as PSM). The context and the usability guidelines established in stage one are taken as input in order to generate a concrete model with the consistent usability guidelines. At a second stage, a model-to code transformation which allows the generation of adaptive UI is processed. Following a detailed explanation of each of these stages.

## 3.1 Stage 1: Selecting Consistent Usability Rules

The aim of this stage is to provide the right usability guidelines for a particular user considering its context information. To achieve this purpose, an intelligent inference system is provided.

A set of usability guidelines specific to our application domain (detailed in the case study section) is identified, initially, and stored in a usability knowledge base. These guidelines are presented in form of production rules. Each with a premise and a conclusion. The context-based usability rules are of the following type: IF *(context$_1$ * context$_2$ * context$_i$)* THEN *usability_guideline*. where *(context$_1$ * context$_2$ * context$_i$)* are called *Premise (set of conditions)* and *usability_guideline* is called *Conclusion*.

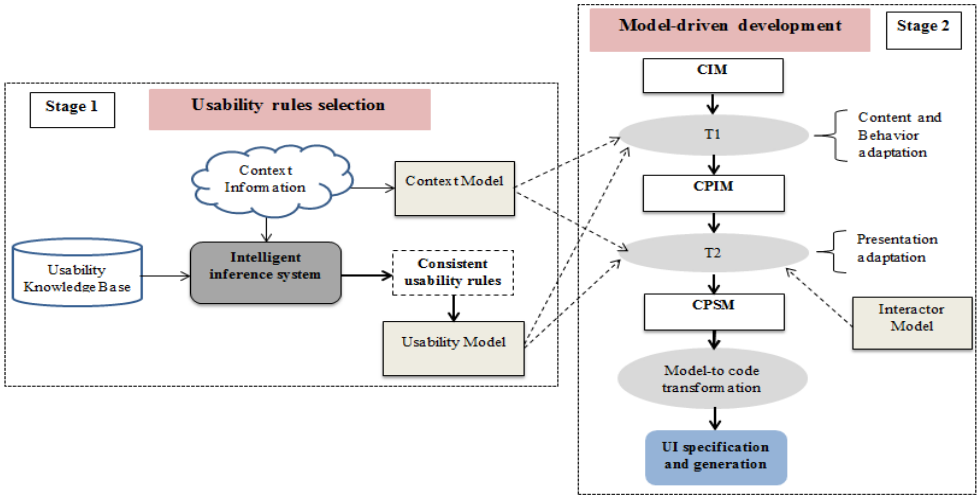


Figure 1: Overview of the proposed approach.

Earlier, the usability engineer attributes a priority index to usability rules with respect to the population characteristics and/or the task requirements.

In experts' experiences, there is always a possibility of conflicts. However, with traditional inference systems, it is difficult to treat conflicting rules and thus the deduction of erroneous knowledge conducts to mistakes in decision making. There may exist varied conflicting rules:

- Conflict between conclusions: For example, a conclusion that provides different steps to inform and guide a user (prompting attribute) and another conclusion that reduce the set of an action steps (brevity attribute). These conclusions contribute to a conflict.
- Conflict between premise and conclusion: For example, *Rule*$_1$: IF environment is noisy THEN display visual notifications (feedback attribute) and *Rule*$_2$: IF user is visually impaired THEN apply vocal mode (flexibility attribute). We can notice that the premise (environment is noisy) has a negative impact on the conclusion (apply vocal mode).

In order to perform the conflict resolution, we adopt a decision-making strategy. Thus, a decision matrix is used to allow the identification of relationships between the set of rules. As we deal with two kinds of conflicting rules, two kinds of decision matrixes are, initially, specified by the usability engineer to indicate whether a conflict between two rules exists (Figure 2):

- Type decision matrix 1: a decision matrix which identifies the semantic relation between different rules conclusions. The elements of the matrix are either zero or one. For example, the usability engineer assigns the value 0, if there is a conflict between the conclusion of the rule 1 (*Rule*$_1$*(Conclusion)*) and the conclusion of the rule m (*Rule*$_m$ *(Conclusion)*), otherwise 1;
- Type decision matrix 2: a decision matrix which identifies the semantic relation between premise and conclusion of different rules. The elements of the matrix are either zero or one. For example, the usability engineer assigns the value 1, if there is a conflict between the premise of the rule 1 (*Rule*$_1$*(Premise)*) and the conclusion of the rule 2 (*Rule*$_2$ *(Conclusion)*), otherwise 0;

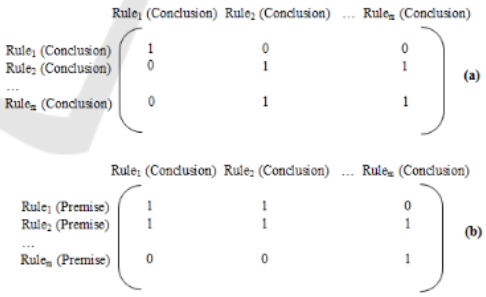


Figure 2: Decision matrixes.

Concerning context instances, a context meta-model is defined at the beginning, which would be exploited in the next stage all along the model-driven development process. By this way, a set of context instances are provided to describe different context situations.

### 3.1.1 Inference System

An inference system is proposed in this section which aims to infer consistent context-based

usability rules. The different steps of our inference system are as follows:

- Step1: Filtering rules
- Step2: Sorting filtered rules
- Step3: Managing conflicts between conclusions
- Step4: Managing conflicts between premises and conclusions.

Next, we detail each of these steps.

**Step 1: Filtering Rules**

A directed graph (or digraph) representation which represents context-based usability rules in order to handle the step of filtering rules is used in this paper. As previously mentioned, a rule is in form of: IF premise THEN conclusion, where the premise is a set of conditions. Figure 3 illustrates the structural dependencies of the digraph. It is constructed by: (1) a set of nodes or vertices; (2) a set of arcs or directed edges: all arcs have arrows that give direction. The graph can only be traversed by the direction of the arrows.

The proposed graph has three levels. The first level identifies the context dimension (i.e. user, environment or platform). The second level identifies all the possible conditions alternatives of rules. The third level includes all the possible conclusions of rules. For each conclusion node, one or a set of conclusion nodes are associated, occurring in the premise part of a rule. Thus, a completed rule is identified by the path from a leaf to a root.

By using a filtering algorithm, we can evaluate the premise with the set of the current contextual parameters and check whether all the condition elements of the premise part in a rule are satisfied. Coming up next is a description of the filtering algorithm:

**Input:** G: digraph, Parameters[]: input contextual parameters.

**Output:** Filtered_List[] : a set of the filtered rules

(1) Iterate and check the context type in the graph G.

(2) Traverse through a sub-graph and check if condition node exist in Parameters[].

(3) Traverse through the satisfied condition node and check if the conclusion node has not yet been tested.

(4) Check if the conclusion node has one or more successors.

(5) Iterate and go to the Step 1 until Parameters[] is totally tested.

**Step 2: Sorting Filtered Rules**

After the filtering rules step and with the purpose of managing the conflict set between rules, the set of the satisfied rules are sorted in order of priority. We first sort the rule with the highest priority index. An existing hybrid algorithm is used as an efficient implementation combining an efficient algorithm for large data sets (the merge sort) with insertion sort for small data sets.

**Step 3: Managing Conflicts Between Conclusions**

In order to make the decision for the consistency of the rules, we used the type decision matrix 1 which allows handling conflicts between conclusions. The steps of this algorithm are: (1) comparing each conclusion of the highly priority rule with the other conclusions of the sorted rules, (2) if the conclusion of the rule j have a negative impact on the conclusion of the rule i, remove the rule j from the filtered list.

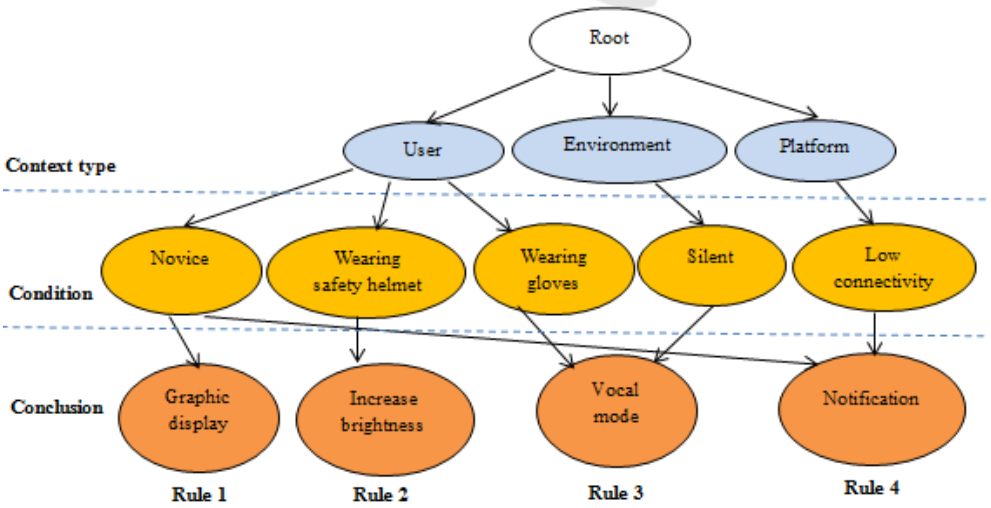


Figure 3: Example of a digraph.

**Step 4: Managing Conflicts Between Premises and Conclusions**

In this step, we used the type decision matrix 2 which allows handling conflicts between premises and conclusions. The steps of the algorithm are: (1) comparing each premise of the highly priority rule with the other conclusions of sorted rules, (2) if the conclusion of the rule j has negative impact on the premise of the rule i, remove the rule j from the filtered list.

At the end of this step, a set of consistent context-based usability rules are provided.

### 3.1.2 Dealing with Usability Guidelines in Model-Driven Transformation Process

In order to be maximized and incorporated in the next stage (the model-driven development process), a usability model which contains the consistent usability rules (previously resulted) should be provided.

The proposed usability model extends the one described in (Harrison et al., 2013). In such model, usability is classified into seven groups and is presented in Table 1.

To define what should be associated with each usability group, we did a detailed literature review about usability modelling. As a result of this review (Table 1), we identified for each usability group the opted usability attributes which we consider appropriate due to their frequent use in the most cited usability model like (Abrahao and Insfran, 2006; Ben Ammar et al., 2015; Scapin and Bastien, 1997; Seffah et al., 2006; Zhang and Adipat, 2005).

In the literature of model transformation, in order to use a model, the definition of the meta-model is a precondition. For that purpose, we define usability meta-model in order to formalize the approach. Figure 4 shows the usability meta-model. It is composed of a *UsabilityGroup*, which defines a set of features used to evaluate the quality of user interface; *UsabilityAttribute*, which is a refined sub-group of the usability group and *UsabilityRule*, which includes *ConditionGroup* describing a set of condition groups (i.e. WHEN clause) and *Conclusion*, defining the usability guidelines (i.e. THEN clause).

In this manner, a usability model, conforms to its meta-model and enables the definition of the context-based usability rules. Using converting algorithms, the set of the consistent usability rules has been converted into the usability model referred to its usability meta-model, in order to be later incorporated in stage 2 (Figure 5).

Table 1: Decomposition of usability groups.

| Group | Attribute | Reference |
|---|---|---|
| Learnability | System feedback | (Ben Ammar et al., 2015) |
| | Grouping | (Scapin and Bastien, 1997) |
| | Legibility | (Scapin and Bastien, 1997) |
| | Prompting | (Ben Ammar et al., 2015) |
| Cognitive Load | Brevity | (Scapin and Bastien, 1997) |
| | Information Density | (Ben Ammar et al., 2015) |
| | Navigability | (Ben Ammar et al., 2015) |
| Satisfaction | Flexibility | (Scapin and Bastien, 1997; Seffah et al. 2006) |
| Error | Error handling | (Seffah et al. 2006) |
| | Error prevention | (Abrahao and Insfran, 2006) |
| Efficiency | Speed of accessing data | (Zhang and Adipat, 2005; Seffah et al, 2006) |
| | Fast access to common tasks | (Abrahao and Insfran, 2006) |
| Effectiveness | Completeness | (Zhang and Adipat, 2005: Seffah, 2006) |
| Memorability | Time to remember | (Zhang and Adipat, 2005; Abrahao and Insfran, 2006) |
| | Accuracy | (Abrahao and Insfran, 2006) |

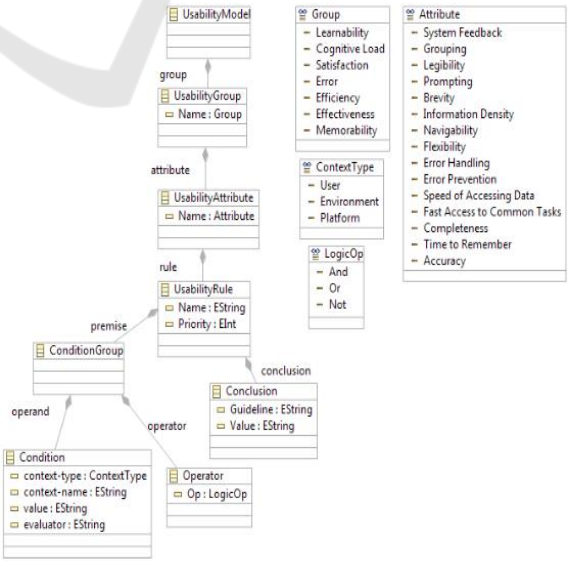


Figure 4: Usability meta-model.

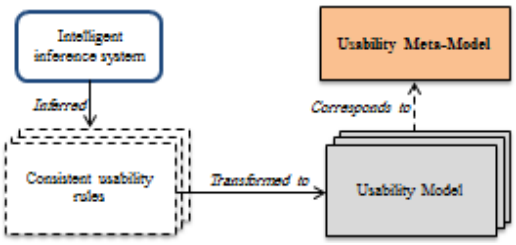


Figure 5: Transformation of consistent usability rules into usability model.

## 3.2 Stage 2: Model-Driven Development

In this stage, we consider the MDA structure. Before presenting the model transformation process, we will explain the main models.

- **Computational Independent Model**

In order to define the CIM model, we use Business Process Modelling Notation (BPMN). As cited in (Brossard et al., 2007), the business process becomes the central part of interaction modelling since it is able to: 1) model all the business goals of the user in an application, 2) define the tasks concerning the specification of each business goal to accomplish it. These tasks are interactive tasks (e.g. data entry), non-interactive tasks and manual tasks, and 3) consider all information flows made between several actors (human or machine) in a single business process.

A better known formalism, the (BPMN, 2006), is chosen which allows the definition of business processes. We justify the use of the BPMN by its ability to model the functional tasks and to describe the business logic of the application. Furthermore, this formalism is especially able to model all information flows between tasks which is particularly important for the integration of the content adaptation. Moreover, additional extensions have been performed to the BPMN. Indeed, each BPMN element in the BPMN model has been annotated with interaction element types in order to describe the type of interaction with the user such as: types of input information, output or grouping information.

- **Usability Model**

Details about usability meta-model have been already explained in the previous section.

- **Context Model**

A defined context meta-model is used during the model transformation process. The context contains a description of user, environment and platform dimension. To identify the information about those elements, a literature review has been fulfilled (Jumisko-Pyykko et al., 2010).

- **Interactor Model**

The interactor model is used during the CPIM-to-CPSM transformation to define design elements of a target interface. It is composed of the interactors which are usually found in tool boxes such as SWING or HTML. A set of containers and contents are specified. The proposed interactor meta-model was initiated by (Sottet, 2008) and then extended.

- **Contextualized Platform Independent Model**

The CPIM model is the output of the first transformation (CIM-to-CPIM transformation) and the input of the second transformation (CPIM-to-CPSM transformation). It allows the description of the user interaction independently of any platform. It is specified in Generic UIML (User Interface Markup Language) meta-model (UIML, 2008). We used UIML as a language for CPIM and CPSM level representation because it allows, in an abstract way, the applications development for the specification of the UI elements. Moreover, we are able to provide the transformation from UIML to source code in diverse platforms (for example, the Eclipse plugin Acceleo allows the conversion to HTML, Java, and Android). The generation of the CPIM model allows the definition of the *structure*, *behavior* and *style* parts of the interface component to be generated.

- **Contextualized Platform Specific Model**

The CPSM model is the output of the second transformation (CPIM-to-CPSM transformation). It is a refinement of the CPIM. Indeed, it allows the description of the user interaction for a specific platform. It is specified in UIML meta-model. The definition of the *presentation*, *logic* and *style* parts of the interface component will be generated.

The model transformation definition consists of a set of transformation rules. We notice that model transformations are implemented with ATLAS Transformation Language (ATL) (ATL, 2006). This language (ATL) allows developers to define transformation rules in order to describe how source model elements are matched and navigated in order to create and produce the target model elements. By applying the CIM to CPIM transformation (T1), the BPM model is associated with usability and context model to allow the content and behavior adaptation. The generation of the *structure*, *behavior* and *style* parts are as follow:

- For the *structure* part, the transformation is accorded to the BPMN element, to which we associate a type of interaction element that describes a specific task. It allows specifying

the interface with the general information. We adopt a generic vocabulary of UI elements, used in conjunction with UIML and which can specify any user interface for any platform (Zaibi et al., 2016). (Ali et al., 2002) provide a more detailed description of it.

- For the *behavior* part, in order to allow content and behavior adaptations, the user interaction is specified with the UI by defining rules. The content adaptation of interactive tasks is applied by using the auto-fill form method for the input information extracted from the context model. The behavior adaptation of interactive tasks is applied by incorporating the usability guidelines.
- For the *style* part, a UIML code is defined manipulating the properties correspondent to content and to specific properties for each UI element, associated to non-interactive tasks (from usability guidelines for graphical UI independently of a particular platform).

By applying the second transformation (T2), a target concrete CPSM model is generated. A set of transformation rules is, thus, established. For each transformation rule, the context, usability and interactor model are taken into account. The *presentation*, *logic* and *style* parts are generated as follows:

- By using the interactor model, the designer can specify each particular platform. The interactor model has a major advantage since it allows developers to avoid implementations at the code generation phase. In this way, the *presentation* part is specified to join generic UIML classes with a specific platform through the interactor model.
- A *logic* part statement will be included containing match between the methods used in the *behavior* part and those defined in the interactor model for a target platform.
- The *style* part is generated, containing content and properties specific for each UI element associated to non-interactive tasks dependently of a specific platform.

**Code Generation.** The last model-to-text transformation is made, to produce the adaptive UI. (Acceleo, 2006) a tool which is built on an MDA based generator, has been supported to accomplish model-to text transformation. Acceleo is chosen because of (1) its adequacy for quickly writing rules for the generation of UI prototype (Model to Text) and (2) its easiness to integrate the existing ATL code in the template.

# 4 CASE STUDY

In order to expose its applicability, we have applied our methodology to an illustrative case study. In this paper, the object of the case study is the work order management. The scenario is the following: Following equipment failure detection, authorized users should be able to connect to the application's database to transfer their work requests. This functionality enables requester, in industrial fields, to create a work request by providing information about it. The system will review the work requests and accordingly either approve or reject the submission. Only approved submissions are converted into work orders and attributed to available staffs into a job process. Then, to proceed on curative interventions, the technician can retrieve the information about the work that has to be done by consulting all information on the work form. After terminating the work, the technician can send a description of the work he processed.

Given that the work order management system is large, we focused our interests on the generation of adaptive UI for the <create work request> and <prepare work orders> tasks. The Business Process Model illustrates our scenarios in the left part of Figure 6.

Different contexts of use have been presented in order to implement our tasks. Table 2 presents the various contexts of use.

Table 2: Example of different contexts of use.

| User profile | User | Environment | Platform |
|---|---|---|---|
| C1: Requester | Wearing gloves | Silent | Tablet |
| C2: Requester | Wearing gloves | Noisy | Smartphone |
| C3: Technician | Novice | In the car | Smartphone |
| C4: Technician | Expert | In the car | Smartphone |

## 4.1 Selecting Consistent Usability Rules

In this study, we are focused on the definition of usability rules regarding the mobility and its consequence. Considering the special circumstances of an industrial environment, we have defined context-based usability rules. An example of usability rules in industrial context-aware environment is presented in Table 3. Beforehand, priority index is attributed by the usability engineer to each usability rule (e.g. R1: (priority=2), R4 and R5: (priority=1)). Besides, two different decision

matrixes are specified by the usability engineer in order to manage the conflict resolution steps.

In order to infer consistent usability rules, the proposed inference system is used by performing the different steps previously explained. The different algorithms are implemented in java.

Once all the different inference system steps have been accomplished, consistent usability rules are resulted. Table 4 shows the consistent usability rules for each particular context of use. We note for example that in context C1, R1 and R5 are conflicting rules since the premise of R5 has a negative impact in the conclusion of the rule R1.

The XML file presented in part (a) of Figure 7 shows the consistent usability rule related to the context C1. In order to be maximized in the model transformation process, algorithms are implemented in java in order to handle XML data of consistent usability rules and retransform them into a new XML-based file which is correspondent to the usability meta-model. Part (b) of Figure 7 presents the usability model.

Table 3: Example of usability rules in industrial fields.

| Usability rule | Signification |
|---|---|
| **R1**: IF platform= smartphone/tablet THEN use tactile interaction | Using tactile interaction with smartphone/tablet platforms. |
| **R2**: IF user = novice and location = in the car THEN provide the GPS functionality | Providing GPS functionality for novice workers in order to get to the work location. |
| **R3**: IF platform = smartphone THEN provide the relevant information. | Providing relevant information in small screen size (e.g. smartphone). |
| **R4**: IF user = wearing protective gloves and noise=high THEN scan the equipment code. | Scanning the equipment code for workers who have got protective gloves and are in noisy environment. |
| **R5**: IF user = wearing protective gloves and noise= low THEN apply vocal interaction. | Switching to vocal mode for workers who have got protective gloves and are in noiseless environment. |

Table 4: Example of consistent usability rules.

| Context | Consistent usability rules |
|---|---|
| C1 | R5 |
| C2 | R4 |
| C3 | R3, R1 |
| C4 | R2, R1 |

## 4.2 Model-Driven Development

This stage fulfils the model transformation process.

### 4.2.1 CIM to CPIM Transformation

A first model transformation T1 allows the generation of the CPIM. The BPM model is in conformance with the BPM meta-model. The execution of the ATL transformation rule is a target CPIM. The CPIM is in conformance with the UIML meta-model. Part (a) of Figure 6 presents the *structure* part of the generated CPIM. Part (b) of Figure 6 presents the *style* part. It describes an example of the integration of usability requirements with a non-interactive task. Parts (c) and (d) of Figure 6 illustrates the *behavior* part for the <create work order> task. These parts show correspondently content (e.g. form auto-filling) and behavior (e.g. usability integration with interactive tasks) adaptation.

### 4.2.2 CPIM to CPSM Transformation

In the second model transformation (T2), a transition is implemented in order to obtain the concrete CPSM considering the context, usability and interactor model. The complete UIML code, which is defined to a specific platform, describes the code generated for the *presentation*, *logic* and *style* parts of the generated CPSM. An example of the generated target model (CPSM) is showed in the right part of Figure 6.

### 4.2.3 CPSM to Code Transformation

Adaptive UI are generated according to Acceleo templates. Figure 8 shows the generated UI for the <create work request> task accordingly to context C1 and C2. The generated UI are web-based. In the case of context C1 (left part of Figure 7), a requester wearing protective gloves and sitting in a noiseless environment is using the application through a tablet. *Flexibility* is considered by switching to vocal mode in order to enable the work request creation. In the case of context C2 (right part of Figure 7), a requester wearing protective gloves, and sitting in a noisy environment is using the application through a smartphone. *Flexibility* is considered by allowing the scan of the code of the malfunctioning equipment.

Figure 9 shows the generated UI for the <process work orders> task. In the case of context C3 (left part), an expert technician, in the car, is consulting the work order information. Added to providing tactile interaction (R1), *Brevity* is considered by providing the relevant information. In the case of context C4 (right part), a novice technician, in the car, is consulting the work order information in order to get to the work location. Added to providing tactile interaction (R1), *prompting* is considered by displaying GPS functionality.

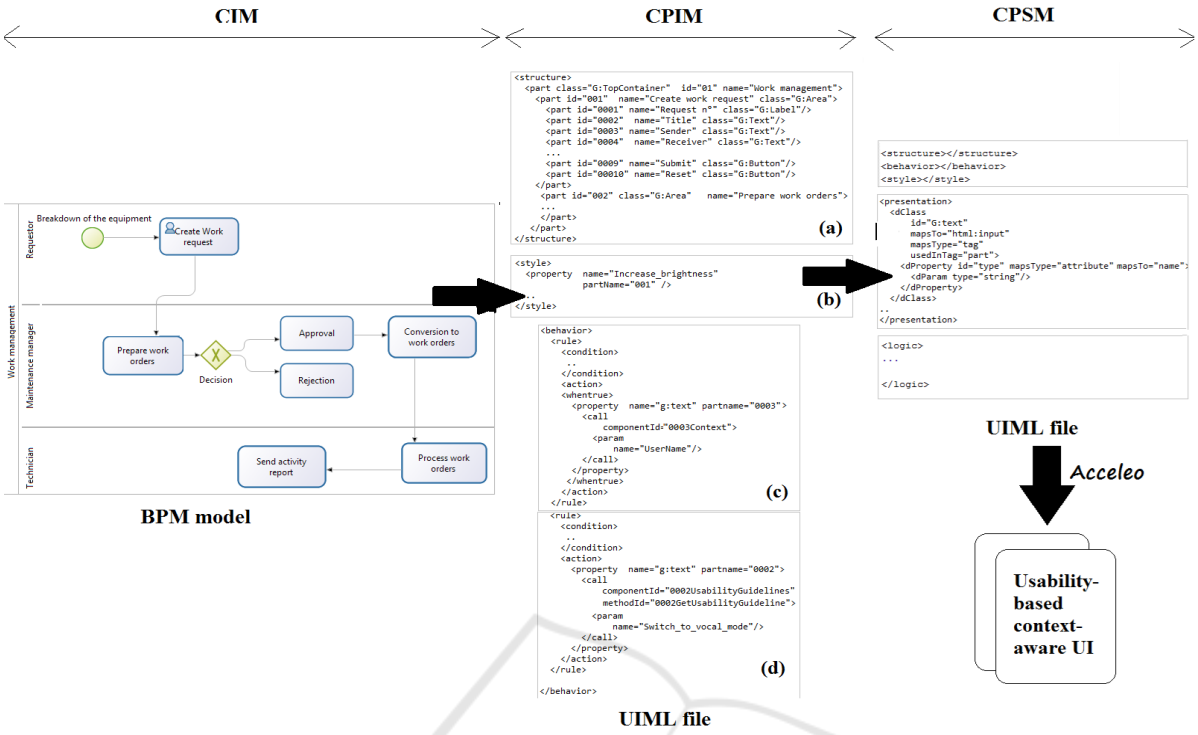


Figure 6: Model-driven transformation process.

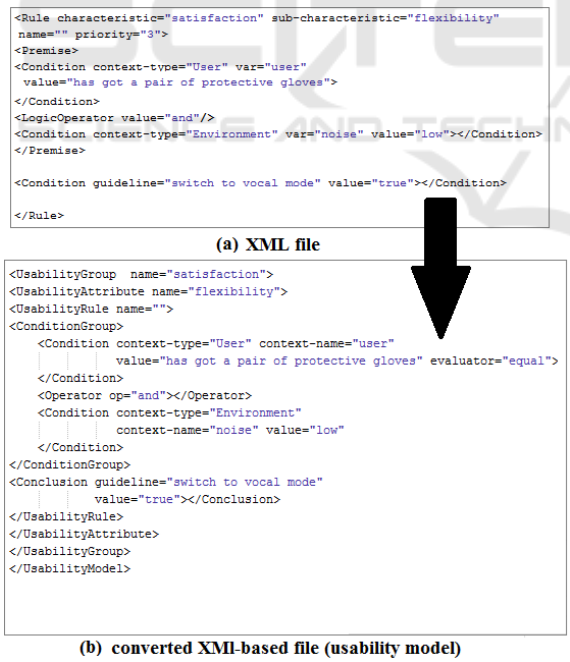


Figure 7: Consistent usability rules: (a) XML file, (b) converted XML-based file.


Figure 8: The generated UI for contexts C1 and C2.

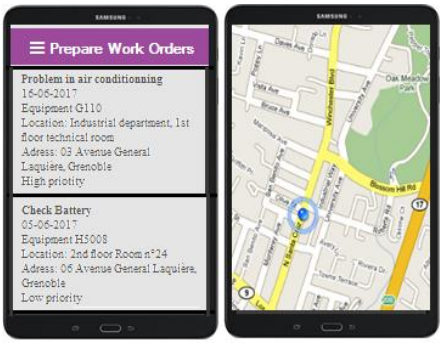


Figure 9: The generated UI for contexts C3 and C4.

# 5 EXPERIMENTS AND RESULTS

In order to validate our proposal, we have evaluated our work management mobile application. We consider a set of experimental subjects. Thirty two participants were invited in the experiment. Table 5 describes the different profiles that participated.

We consider a questionnaire to gather end-users' opinions and perceptions. A twenty question (ISO, 2010; Bastien and Scapin, 1997) ergonomic criteria based questionnaire is handed to the participants. This questionnaire considers the following criteria (*adaptation*, *ease of use*, *efficiency*, *effectiveness*, and *satisfaction*) as presented in Table 6. The final results are illustrated in Figure 10.

According to these results, we can conclude that the majority of the participants are satisfied (87.5%). This is due to the quite satisfaction of their requirements with regard to the changing context of use. Concerning the *efficiency*, almost 90% of the users affirmed that they accomplished the work rapidly. This is due to the easy access to the system functionalities. Furthermore, 81% of the users asserted the adaptive UI ease of use.

Despite of the participants' heterogeneity and the highly dynamic context of use, almost all users asserted that the use of the adaptive application was very effective in term of reaching targets and responding to users' preferences and requirements.

Table 5: Profiles of participants.

| Profile | Age | Number | User experience |
|---|---|---|---|
| Profile 1 | 30-35 | 8 | Novice |
| Profile 2 | 35-40 | 8 | Expert |
| Profile 3 | 40-45 | 8 | Novice |
| Profile 4 | 45-50 | 8 | Expert |

Table 6: Defined criteria questions.

| Criteria | Question |
|---|---|
| Efficient | Can the user finish the work quickly during the interaction with the interface? |
| Effectiveness | Have you achieved what you intended to do with the interface? |
| Adaptation | Are your requirements sufficiently taken into account while exploiting the interface? |
| Satisfaction | Working with the interface is satisfying? |
| Ease of use | The interface manipulation is easy? |

According to the users' responses, we have perceived that: (1) manipulating the adaptive system is greatly efficient and the users have quickly accomplished their objectives, (2) the users' objectives were effectively, (3) the system adapted to the users profiles in different ways, (4) the users declared satisfaction for using the adaptive system and, (5) the users can use the adaptive system easily.

As researchers, we concluded that the proposed context-aware application ensures an optimal degree of usability. Changing the context of use will allow the variation of the system's behaviour and presentation. Our proposed adaptive system will enable users to support activities, anticipate their needs without assistance and experience with more independence while working and communicating too.

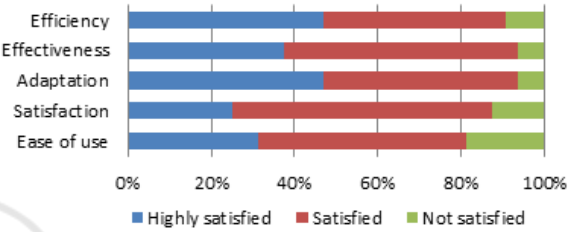


Figure 10: The system evaluation: a questionnaire.

# 6 CONCLUSION

This paper presented an approach that addresses context-awareness and usability issue as parts of an adaptive UI development process. The main motivation of the contribution is to combine these issues in a conformity process to provide a more reliable design of UI. The proposed methodology is defined through two stages namely: the selection of consistent usability rules and the model-driven development. Firstly, our proposal is based on an inference system for deducing consistent usability rules with regard to the context of use. Secondly, a model-driven transformation process has been fulfilled in order to incorporate both context-awareness and usability requirements into the Model-Driven Architecture.

With regard to the existing proposals, the usability-based context-aware concept initiated in this paper presents the following benefits: (1) usability requirements are consistently selected in context-aware environments; (2) usability of context-aware systems is processed since an early design stage; and (3) experiments with end-users have been carried out.

The continuity of our research work leads directly to the investigation of more context-based usability rules with the perspective of the users in

order to depict other requirements and to continue the evaluation of our approach and incorporate findings into future enhancements.

## ACKNOWLEDGEMENTS

This project has been developed in the scope of a MOBIDOC doctoral thesis of the PASRI program financed by the European Union and administered by the ANPR.

## REFERENCES

Abrahao, S. M., and Insfran, E., 2006. Early usability evaluation in model driven architecture environments. In *QSIC*, pages 287–294.

Ali, M.F. Perez-Quinones, M.A., Abrams, M., and Shell, E. 2002. Building multiplatform user interfaces with uiml, *In Proceedings of Computer Aided Design of User Interfaces*, pages 255-266.

Aquino, N., Vanderdonckt, J., Condori-Fernendez, N., Tubio, O. D., and Pastor, O., 2010. Usability evaluation of multi-device/platform user interfaces generated by model-driven engineering, *ESEM*

ATLAS group LINA & INRIA, ATL, Atlas Transformation Language, 2006. *ATL User Manual - version 0.7*

Ben Ammar, L., Trabelsi, A., and Mahfoudhi, A., 2015. Incorporating usability requirements into model transformation technologies, *Requirement Engineering*, 20(), pages 465-479.

Ben ayed, E., 2017. *Une approche pour l'évaluation des systèmes d'aide à la décision mobiles basés sur le processus d'extraction des connaissances à partir des données : Application dans le domaine médical*. PhD thesis, National school of Engineers of Sfax and University of Valenciennes and Hainaut-Cambrésis.

Brossard, A. Abed, M., and Kolski, C., 2007. *Modélisation conceptuelle des IHM : Une approche globale s'appuyant sur les processus métier*, Ingénierie des Systèmes d'Information (ISI) – Networking and Information Systems, vol 12, pages 69-108.

Dey, A.D., and Abowd, G.D., 2000. Towards a Better Understanding of Context and Context-Awareness, *CHI Workshop on the What, Who, Where, When, and How of Context-Awareness*.

Eclipse Acceleo, *https://eclipse.org/acceleo/*, last visited on 30 July 2017

Gonzalez-Huerta, J. Blanes, D. Insfran, E., and Abrahao, S., 2010. Towards an Architecture for Ensuring Product Quality in Model-Driven Software Development, *PROFES*

Harrison, R., Flood, D., and Duce, D., 2013. Usability of mobile applications: literature review and rationale for a new usability model, *Journal of Interaction Science*.

Hentati, M., Ben Ammar, L., Trabelsi, A., and Mahfoudhi A., 2016 An Approach for Incorporating the Usability Optimization Process into the Model Transformation, *ISDA.*

ISO: ISO 9241-210, 2010, *Ergonomics of human-system interaction*

Jumisko-Pyykko, S., and Vainio, T., 2010. Framing the Context of Use for Mobile HCI, *International Journal of Mobile Human Computer Interaction*, 2(4), pages 1-28

Oliveira, K.M., Bacha, F., Mnasser, H., and Abed, M., 2013. Transportation ontology definition and application for the content personalization of user interfaces, *Expert Systems with Applications*, 40(8), pages 3145-3159.

OMG, BPMN – Business Process Modeling Notation Specification version 1.0, 2006. *OMG Available Specification.*

OMG, MDA Guide Version 1.0, 2003, *http://omg.org/mda/mda_files/MDA_Guide_Version1-0.pdf*

Ormeno, Y. I. ,Panach, J. I., Condori-Fernandez, N., and Pastor, O., 2013. Towards a proposal to capture usability requirements through guidelines, *RCIS.*

Panach, I. J., Aquino, N., Pastor, O., 2014. A proposal for modelling usability in a holistic MDD method. *Sci. Comput. Program. 86*, pages 74-88

Paterno, F., Carmen, S., and Lucio, D. S., 2009. MARIA: A universal, declarative, multiple abstraction-level language for service-oriented applications in ubiquitous environments. *TOCHI*. 16(4),

Pathan, K.T., and Reiff-Marganiec, S. 2009. Towards Activity Context using Software Sensors, *YR-SOC*, pages 27-35

Scapin, D. L., and Bastien, J. M. C., 1997. Ergonomic criteria for evaluating the ergonomic quality of interactive systems, *Behaviour & Information Technology*, 16(4), pages 220-231.

Seffah, A., Donyaee, M., Kline, R. B., and Padda, H. K. 2006. Usability measurement and metrics: A consolidated model, *Software Quality Control*, 14(2), pages 159–178.

Serral, E., Valderas, P., and Pelechano, V., 2010. Towards the model driven development of context-aware pervasive systems, *Pervasive and Mobile Computing*, pages 254-280.

Sottet, J. S. 2008. *Mega-IHM: malléabilité des Interfaces Homme Machine dirigées par les modèles*, PhD Thesis, University of Joseph Fourier.

User Interface Markup Language (UIML), Version 4.0. Oasis, 2008, *http://docs.oasisopen.org/uiml/v4.0/cd01/uiml4.0- cd01.html*

Zaibi, D., Riahi, M., and Moussa, F., 2016. Formalization of ergonomic knowledge for designing context-aware human-computer interfaces, *ICSEA*

Zhang, D. and Adipat, B., 2005. Challenges, methodologies, and issues in the usability testing of mobile applications, *International Journal of Human-Computer Interaction*, 18(3), pages 293-308