A Model-based Approach for Mixed-Initiative Context-aware Adaptation of Graphical User Interfaces

By Nesrine Mezhoudi

Thesis submitted in the fulfillment of the requirements for the degree of

Doctor of Philosophy
in Engineering Science
of the École Polytechnique de Louvain
Université catholique de Louvain

Committee in charge:

Prof. Gaëlle Calvary, Grenoble Institute of Technology, Examiner
Prof. Manuel Kolp, Université catholique de Louvain (ICTeam), Reader
Prof. Olfa Daassi, Université de Tunis, Examiner
Prof. Vivian Genaro Motti, George Mason University, Virginia, Reader
Prof. Christophe DeVleeschouwer, Université catholique de Louvain (ICTeam), President
Prof. Jean Vanderdonckt, Université catholique de Louvain (ILSM & ICTeam), Advisor

June 21th 2016
Albert Einstein is attributed the famous inspirational quote, “Computers are incredibly fast, accurate, and stupid. Human beings are incredibly slow, inaccurate, and brilliant. Together they are powerful beyond imagination.”
First and foremost, I would like to express my sincere gratitude to my advisor Prof. Jean Vanderdonckt for his continuous support to my Ph.D study and related research, for his motivation, and immense knowledge. His guidance helped me all along the research and writing time of this thesis. I could not have imagined having a better advisor and mentor for my Ph.D study. Besides my advisor, I would like to thank all the members of the jury: Prof. Gaëlle Calvary, Prof. Manuel Kolp, Prof. Vivian Motti, Prof. Olfa Daasi and Prof. Christophe De Vleeschouwer.

Getting through my dissertation required more than academic support, and I have to thank:

- All LILab people for their friendship, support and collaboration: Vivian, Diane, Iyad, Ugo, Mathieu, Jorge, Pascal, Nicolas, Suzanne, François et Jeremy.
- Doyen and UCL peoples: Marco, Nathalie, Yvonne, Sylvie, Francoise, Sabine, Hassan, Samedi, Mehdi, Wenli and Mme De Potter.
- Namur friends: Hajer et Nesrine.
- I would also like to thank Iyad and Hana who opened both their home and heart to me when I first arrived in the city.

A special thanks to my family. Words cannot express how grateful I am to my mother, and father for all of the sacrifices that you’ve made for me. Your prayer for me was what sustained me so far.

I would also like to thank my dear brothers (My Heroes): Nizar, Khaled and Khalil for their support.

At the end I would like to express appreciation to my beloved husband, Moez, who has been my best friend and great companion. He loved, supported, encouraged, entertained, and helped me get through this agonizing period in the most positive way.

I would like to dedicate this thesis to the memory of my brother, Nader Mezhoudi (1990-2011), who is a constant source of inspiration. I hope he would have been proud.
This dissertation was carried out thanks to the support of:

- EU FP7 Serenoa project (Multidimensional context-aware adaptation of Service Front-ends) funded by the European Union through its Seventh Framework Program as a STREP Project no. FP7-ICT-258030
- EU ITEA2-Call 8 UxiXML (User Interface eXtensible Markup Language) project funded by Direction Générale DGO6 of Région Wallonne under EU label n°2008026
- Jounum project that is funded by Direction Générale DGO6 of Région Wallonne under convention n°1317963
Abstract

Context-aware adaptation is the key to the success of a user interaction and a support for usability. This fact is recognized and empirically assessed by the HCI community. A significant amount of research has been conducted to design and develop context-aware adaptation in interactive applications in order to streamline the user interface development lifecycle. It has to be noted that adaptations could lead to the end-users’ frustrations and the interaction downfall when they do not meet the users’ needs and expectations. Thereby ‘Context-awareness’ as well as ‘user-centeredness’ become crucial to improve the quality of interaction. In this sense, adapting user interfaces ‘intelligently’ in the executing environment in response to context changes, such as: location, resources and platform rather than fulfilling user preferences and needs, enhances the interaction. Nevertheless, most interface adaptations are managed at design time, instead of corresponding to the situation and the ambient-context. An accurate and successful adaptation should be context-aware, user-centered and have a crosscutting impact on software patterning and appearance at runtime, with an insignificant cost. This is what this thesis proposal is aimed at.

In order to address the aforementioned shortcomings of bridging the gap between adaptation goals and user expectations, this thesis provides a methodology for developing mixed-initiative adaptive user interfaces. It supports an agile runtime adaptation capitalizing on user interventions. It outlines an approach that conveys an extensive characterization and covers the entire decision-making process as regards a runtime mixed-initiative context-aware adaptation. Along with practical implementation guidance, two support tools instantiating the methodology were developed for the generation of a mixed-initiative user interfaces. Diverse instantiations are proposed, exploiting Machine Learning (ML) potential to improve UI intelligibility considering the user preferences at runtime.

Keywords: Runtime Mixed initiative Context-aware Adaptation; Human-Computer Interaction; Machine learning, User feedbacks
TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION ........................................... 14

1.1 Context of the problem .................................................. 14
  1.1.1 Definitions ......................................................... 15
  1.1.2 Motivations ......................................................... 17

1.2 Thesis Statement .......................................................... 19
  1.2.1 Research questions ................................................. 19
  1.2.2 Thesis statement .................................................... 20
  1.2.3 Aims and Scope ..................................................... 21

1.3 Thesis Outline ............................................................. 22

CHAPTER 2 RELATED WORK ............................................. 25

2.1 Adaptation in HCI .......................................................... 25
  2.1.1 Adaptation based on context information ....................... 26
  2.1.2 UI Adaptation: Granularity of changes ......................... 32
  2.1.3 Approaches to UI Adaptation: a user involvement continuum ............................................ 34
  2.1.4 Discussion .......................................................... 40

2.2 Support for MICAA .......................................................... 41
  2.2.1 Machine Learning techniques ..................................... 41
  2.2.2 Techniques relevant for HCI ....................................... 43
  2.2.3 Machine Learning Applied to HCI ................................ 44
  2.2.4 Discussion and final remarks ...................................... 52

2.3 Agile practices meeting adaptation requirement ...................... 56
  2.3.1 Agile methods: Overview ........................................... 57
  2.3.2 Adaptation requirement ............................................ 59
  2.3.3 Discussion .......................................................... 61

2.4 MICAA Quality ............................................................. 63
  2.4.1 Quality and Adaptation approach ................................ 64
CHAPTER 3  MICAA FRAMEWORK  ................................................. 71

3.1  UFOnt: User Feedback Ontology ........................................... 72

3.2  MICAA Conceptual Framework .............................................. 78
  3.2.1  MICAA Meta-model (MICAA-MM) ................................... 79
  3.2.2  MICAA Structure (MICAA-str) ....................................... 84
  3.2.3  MICAA Methodological framework (MICAA-MF) ............... 88

3.3  MICAA Interaction Quality Support (MICAA-IQS) .................... 93

CHAPTER 4  IDENTIFICATION OF THE ABSTRACT-USER INTERFACE  ................................................. 97

4.1  Definitions and analysis of the sub-problem ............................. 97
  4.1.1  The problem of AUI identification ................................. 97
  4.1.2  Exploration tracks ..................................................... 98
  4.1.3  Motivations for AUI identification ................................. 100

4.2  The proposed solution: M@rina ........................................... 101
  4.2.1  Algorithms ............................................................... 104
  4.2.2  M@rina: Support tool .................................................. 114
  4.2.3  M@rina: Evaluation ..................................................... 117

CHAPTER 5  USER INTERFACE WIDGET SELECTION  124

5.1  Definitions and analysis of the sub-problem ............................. 124
  5.1.1  The problem of widget selection .................................... 124
  5.1.2  Existing solutions ...................................................... 125
  5.1.3  Motivations for context-aware widget selection ................. 126

5.2  The proposed solution: WISEL ........................................... 128
  5.2.1  Algorithms ............................................................... 132
Table of Figures

Fig. 1. a) One single swipe of a hand is enough in the Magic User Interface by Continental to move content from one screen to another [Continental 2011]. b) Users picking up objects from the table at a public demo of LightSpace. Picking up objects from the table is accomplished by swiping them into one’s hand; following the pick-up, one can see an iconic representation of the object (a ball) in their hand [Wilson et al. 2010].

Fig. 2 Thesis contributions

Fig. 3 Reading Map and thesis structure

Fig. 4 Adaptation granularity from Minor to Major change.

Fig. 5 A continuum of user involvement for adaptation approaches.

Fig. 6. Existing works coverage on the adaptation continuum.

Fig. 7 A task grouping sample [Mezhoudi 2013].

Fig. 8 Potential Multiple-choice widgets definition for a known domain.

Fig. 9 Project types supported by agile processes [Vijayasarathy et al. 2008].

Fig. 10 MICAA Conceptual framework.

Fig. 11 A user's Feedback Ontology [Mezhoudi et al. 2015].

Fig. 12 Interaction of users goal and feedback specificity [Ilgen et al. 2019].

Fig. 13 Extended classification of behaviors that can be used for implicit feedback from [Oard et al. 2001]. Sample of possible behaviors that user might exhibit.

Fig. 14 Knowledge based Human computer Interaction [Fisher 2001].

Fig. 15 Adaptation main concepts.

Fig. 16 Unified Adaptation model for runtime context-awareness [Mezhoudi et al. 2015].

Fig. 17 The User Model.

Fig. 18 The Platform Model.

Fig. 19 The Environment Model.

Fig. 20 Method for supporting the analysis of context-aware Adaptation.

Fig. 21 Agile methodology loops for adaptation support.

Fig. 22 Agility supporting UI adaptation process.
Fig. 23 Agile Adaptation in the support of OMP [Vanderdonckt et al. 07] .... 91
Fig. 24 Practical Guidance ................................................................. 91
Fig. 25 Context-Aware Intelligent Interaction Architecture: based on user’s cognitive model and usability features ........................................ 95
Fig. 26 MICAA based AUI identification: M@rina instantiation .............. 102
Fig. 27 M@RNA partial AUI Reification .................................................. 103
Fig. 28 M@RNA GUI for supporting controllability: It consist on a list of recommended AUI ordered with regard to the scoring function (AUI profit) and in the main screen a configuration screen allowing the tuning of the list of parameters considered by scoring function ............. 105
Fig. 29 a) Explicit feedback implementation: rating scale. b) Classes managing the user feedback: Partial view ........................................ 108
Fig. 30 Input-Output calculating score module ....................................... 112
Fig. 31 The simulation of reification process ........................................... 113
Fig. 32 a) The MoneyTransfer Task model. b) First autonomous adapted UI: (user-feedback independent). ...................................................... 115
Fig. 33 a) Reification classes Scenario accomplishing a classic transfer. b) Scenario of an IBAN transfers realization. c) Scenario invoking the controllability feature ................................................................. 116
Fig. 34 Computing LA cost for the generated interface layout .................... 118
Fig. 35 a). Average Time-consuming by users. b). Average Time-consuming tasks for different iterations .................................................. 121
Fig. 36 Tukey: Difference between group means ..................................... 123
Fig. 37 MICAA based CUI selection: WiSel instantiation ....................... 129
Fig. 38 Widget selection process: Three adaptations directed support the adaptation process, the designer knowledge, the user direct manipulations and the advanced logic based on the scoring function .... 131
Fig. 39 WiSel recommendation feature class model .................................. 134
Fig. 40. Implicit UF: WiSel log file ⊕, displayed User’s profile vectors ⊙ .... 137
Fig. 41 A. Widget personalization of ‘Specify color’ interactive tasks B. Form for delineating the scoring functions ........................................ 138
Fig. 42 The scoring function configuration .............................................. 138
Fig. 43 Car reservation task tree ............................................................. 139
Fig. 44 Widget personalizations of ‘Specify color’ interactive tasks .......... 140
Fig. 45 Paradigms for incorporating context RS [Adomavicius et al. 11] ........ 147
Fig. 46 MICAA based recommendation: JouNum instantiation .......... 149
Fig. 47 Potential behaviors of consultation .................................... 153
Fig. 48 JouNum: Context-aware news ranking ............................. 155
Fig. 49 The Business Canvas Model ........................................... 162
Fig. 50 MICAA Business Canvas Model ...................................... 162
# Table of Tables

Table 1. Characteristics of explicit and implicit feedback for interactive systems [Jawaheer et al. 10] .................................................................................................................................................. 30  
Table 2. Identified criteria for the Characterization of adaptation approaches ........................................................................................................................................................................................................................................ 35  
Table 3. ML techniques classification: Overview ........................................................................................................................................................................................................................................ 42  
Table 4. Roadmap of AI techniques for the support of HCI [Motti et al. 2013] .................................................................................................................................................................................................................................................................... 44  
Table 5. Characterization of adaptation approaches based on [Roth 13] using Harvey balls (poor Good support) ........................................................................................................................................................................................................................................ 51  
Table 6. Definition of 12 Agile principal [Beck et al. 2001] ........................................................................................................................................................................................................................................ 58  
Table 7. Differences between agile development and classical development methods and life cycles .................................................................................................................................................................................................................................................................... 59  
Table 8. Similarities between Agile Method basis and HCI Practices .................................................................................................................................................................................................................................................................... 63  
Table 9. Description of structure layers ........................................................................................................................................................................................................................................................................ 86  
Table 10. Analysis of methodological stages ........................................................................................................................................................................................................................................................................ 92  
Table 11. Association between requirements and respective Design Decisions taken ........................................................................................................................................................................................................................................................................ 94  
Table 12. ANOVA test results on average rate of Workload consuming 121  
Table 13. ANOVA test results on average rate of Workload consuming for explicitly and implicitly adapted sessions .................................................................................................................................................................................................................................................................... 122  
Table 14. ANOVA test results on average rate of Workload consuming for explicitly and implicitly adapted sessions .................................................................................................................................................................................................................................................................... 122  
Table 15. Potential scoring equations ........................................................................................................................................................................................................................................................................ 135  
Table 16. Confusion matrix of “gender selection” interactive task ........................................................................................................................................................................................................................................................................ 142  
Table 17. Pilot user study results ........................................................................................................................................................................................................................................................................ 143  
Table 19. A characterization of MICAA support for recommendation challenges using Harvey balls (poor Good support) .................................................................................................................................................................................................................................................................... 151  
Table 20. Requirement Coverage visualization using Harvey balls (poor Good support) .................................................................................................................................................................................................................................................................... 159
# Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACM</td>
<td>Association for Computing Machinery</td>
</tr>
<tr>
<td>AEHS</td>
<td>Adaptive Educational Hypermedia Systems</td>
</tr>
<tr>
<td>AI</td>
<td>Artificial Intelligence</td>
</tr>
<tr>
<td>AM</td>
<td>Agile Methodology</td>
</tr>
<tr>
<td>AUI</td>
<td>Adaptive User Interface</td>
</tr>
<tr>
<td>CAA</td>
<td>Context-Aware Adaptation</td>
</tr>
<tr>
<td>CHI</td>
<td>Computer-Human Interaction</td>
</tr>
<tr>
<td>CRF</td>
<td>Cameleon Reference Framework</td>
</tr>
<tr>
<td>DT</td>
<td>Decision Tree</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>HCI</td>
<td>Human-Computer Interaction</td>
</tr>
<tr>
<td>HTA</td>
<td>Hierarchical Task Analysis</td>
</tr>
<tr>
<td>IAUI</td>
<td>Intelligent Adaptive User Interface</td>
</tr>
<tr>
<td>IS</td>
<td>Information System</td>
</tr>
<tr>
<td>IUI</td>
<td>Intelligent User Interface</td>
</tr>
<tr>
<td>LA</td>
<td>Layout Appropriateness</td>
</tr>
<tr>
<td>LRS</td>
<td>Longest Repeating Subsequence</td>
</tr>
<tr>
<td>MBUI</td>
<td>Model-Based User Interface</td>
</tr>
<tr>
<td>MDE</td>
<td>Model-driven Engineering</td>
</tr>
<tr>
<td>ML</td>
<td>Machine Learning</td>
</tr>
<tr>
<td>MIA</td>
<td>Mixed-Initiative Approach</td>
</tr>
<tr>
<td>MICAA</td>
<td>Mixed-Initiative Context-Aware Adaptation</td>
</tr>
<tr>
<td>SE</td>
<td>Software Engineering</td>
</tr>
<tr>
<td>SDLC</td>
<td>Software Development Life Cycle</td>
</tr>
<tr>
<td>UBP</td>
<td>User Behavior Prediction</td>
</tr>
<tr>
<td>UI</td>
<td>User Interface</td>
</tr>
<tr>
<td>UIDL</td>
<td>User Interface Description Language</td>
</tr>
<tr>
<td>UF</td>
<td>User Feedback</td>
</tr>
<tr>
<td>UsiXML</td>
<td>User Interface eXtensible Markup Language</td>
</tr>
<tr>
<td>UX</td>
<td>User Experience</td>
</tr>
<tr>
<td>W3C</td>
<td>World Wide Web Consortium</td>
</tr>
<tr>
<td>WP</td>
<td>Workload Profile</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
</tbody>
</table>
Chapter 1  Introduction

1.1 Context of the problem

With the rise of modern hardware technologies used in every area of daily life and their widespread accessibility, such as smartphones and tablets, there is also an increasing demand for more efficient and effective software. Hence an incremental need arises for interactive systems that support diverse interaction requirements and appropriately fits the users’ needs through what the technology is able to afford. There is still a gap between user expectations and designed UIs preventing the efficient use of interactive systems.

*Context-aware adaptation* is aimed at reducing such a gap by adjusting the diverse interface aspects with respect to the context of use, which consists of any information that is relevant for the characterization of the interaction including users, the platform and the environment [Calvary et al. 03].

Most of the existing User Interfaces (UIs) are designed for a conventional context of use: an able-bodied user interacting via a desktop computer in a static environment. However, designing UIs with regard to a changing context of use is no longer a luxury but a necessity. It is supposed to improve the interaction and support the task’s achievement. Most of existing approaches consider conventional context and do not comply with the continuous changes of the ambient context. Such adaptation approaches are not suitable for the real interaction scenario and do not fit user’s expectations and their current requirements. Besides it prevents them to build their UIs according to their preferences and limit their effectiveness. We support adaptation approaches that take into account the user interaction with the interface in runtime to decide adaptation, such user-centered perception orient toward helping users to achieve their goals besides decreasing the required workload [Norcio et al. 89][Lieberman 09]. Several works focus on a context-aware user-centered adaptation paradigm. They consider end-user needs throughout acquired information from experimental user studies and user
modeling. The main challenge targeted by current researches in HCI is to develop context-aware interactive systems that have the ability to offer an effective adaptation fitting runtime context requirement [Ralph et al. 80].

1.1.1 Definitions

The adaptation of user interfaces has been extensively investigated by the HCI community resulting into a huge amount of literature pieces and case studies. For example, UI adaptation has been defined as a set of changes that occurs when certain contextual characteristics are identified and used to provide purposeful UI transformations [Browne et al. 86].

Adaptation. This process is decomposed into two categories depending on which entity is controlling the adaptation: adaptability refers to the capability of the user interface to be adapted by the end-user (hence, the Latin suffix “able”), whereas adaptivity refers to the capability of the user interface to adapt itself to the user, to his/her task, and to the interaction between the user and the system. Previous research has studied the benefits of letting users adapt the UI to their own preferences and tasks [Rosson 84, Oppermann 94, Frias-Martinez et al. 09]. Mixed-initiative has been introduced [Horvitz99] to make both the end-user and the system collaborating in the adaptation process. Adaptation to the context of use is a proven solution to improve interaction and to contribute in reducing the gap between the users expectations and the systems’ decisions. However predesigned adaptation cannot offer well-timed, desirable and context-aware customization of interaction efficiently. Accordingly adaptation needs to consider with care the execution context of use so that the resulted customizations are not only suitable for the interaction context but also correspond to the real need of end-users. Several studies investigated various UI adaptation dimensions [Dieterich et al. 93], i.e. what constitutes an adaptation, the level and timing of adaptation, the controlling agent, the required knowledge to reach meaningful adaptations, etc.

Adaptability. The challenges of designing and building adaptable interfaces have produced a rich and varied literature. For instance, multi-layered interfaces [Shneiderman 03] are aimed at identifying the most frequently used UI actions and to emphasize them so as to facilitate their access through a dedicated layer. Sophisticated techniques have been proposed to let end users adapt their UI at run time, even without access to the source code. The End-User Development (EUD) is an area that definitely looks at means to ensure capabilities like adaptability, but not only. Yet, powerful adaptable approaches generally require users to spend significant time and
Chapter 1. Introduction

effort adapting their UIs, often by writing scripts or editing code. Consequently, these approaches have not yet been widely adopted in widely disseminated interactive systems.

**Adaptivity.** Adaptive approaches attempt to address customization by automatically creating and maintaining various models (e.g., a user model, an activity model, an interaction trace) to predict and trigger potentially adequate adaptations. Mostly the adaptive approach considers prior interaction knowledge, instead of capitalizing more on the end-users interventions for endorsing adaptation decisions through their commitment and validation. In the other hand, transferring the total control of the adaptation to an automated system has significant drawbacks, such as permanent change of the UI (which can induces a mental rupture in the end user’s perception and increase his/her workload), unpredictability of the adapted UI, and lack of consistency across adapted versions [Jameson et al. 03, Hook et al. 99, Keeble et al. 00, Shneiderman et al. 97].

**Mixed-initiative.** These approaches can alleviate those problems by suggesting possible adaptation to the end user, who remains in full control of the adaptation process. Ultimately, adaptive and mixed-initiative approaches put the focus on developers to build and integrate complex user models into the UI, which is difficult to do well [Bunt 07]. For these reasons, we hypothesize in this thesis that mixed-initiative adaptation should be explicitly addressed in order to support UI context-awareness. The major challenge is to involve the end user to gather useful knowledge from runtime interactions, in order to increase the performance of the adaptation. User feedback and runtime interventions may be regarded as “input amplifiers”. They enable the users to state their intent at a high level and translate it into low-level recommendations needed to refine the adaptation. User feedback at its most basic levels, is information provided by users about his/her preferences and/or goals. It gives information on the accuracy, adequacy, correctness, and appropriateness of the system state. As every user interaction can contribute to an implicit interest-indicator [Claypool et al. 01], many researches focus on to the unconscious interaction as useful data for adaptation [Leiva et al. 1, Eisenstein et al. 00, Lee et al. 06]. On the other hand, more truthful knowledge for adaptation could be acquired by involving the user during runtime in an explicit way, which shows generally more expressivity than implicit feedback [Amatriain et al. 09]. The literature offers several techniques to gather valuable user feedback during interaction [Arhippainen et al. 04]. User interventions provide the potential to move part of this feedback from design time to run time. Since users’ needs change continuously, building in an adaptive ability for
Chapter 1. Introduction

the UI gives the system flexibility as the need arises and improve the context-awareness of adaptation without just having to wait for the next software’s version.

**Context-aware adaptation.** To unleash the full potential of mixed-initiative in the adaptation process, the UI adaptation should be conducted with respect to any parameter that may effectively and positively influence the UI adaptation. Parameters could come from the user and her interactive task, the platform used by the end user, and the environment in which the interactive task is conducted on the platform. For these reasons, we assume in this thesis that mixed adaptation should be aware of the context of use. Many definitions [Browne et al. 86, Fischer et al. 12, Brusilovsky 01] of the context of use exist, the most frequently quoted being [Dey et al. 00]: the context of use consists in any information that is relevant to define the behavior of an application, including mainly but not only, information about the user, the platform and the environment. In the remainder of this thesis, the context of use will be defined as a triplet C= (U,P,E), where U refers to a user model, P to a platform model, and E to an environment model [Calvary et al. 03]. Consequently, a Mixed-Initiative Context-Aware Adaptation (MICAA) is a promising avenue for defining an effective approach for supporting UI adaptation, namely by avoiding restricting it to system only.

While combining these approaches reveals some potential benefits, there is still a lack of support tools and applications that are applicable to a large range of mainstream interactive applications. Some really adaptive UIs have been developed so far, among them are [Lieberman et al. 01, Liu et al. 03, Ivory et al. 02, Eisenstein et al. 00, Langly et al. 97] but MICAA approach is expected to deliver more potential.

**1.1.2 Motivations**

More skilled software is not necessarily easier to use. More gadgets sometimes cause more complications. What can we do to make sure that the increased capability of our artifacts actually improves users’s lives? Expanded UIs assume and support context-aware interaction by adjusting some interface aspects with regard to the interaction context such as the screen size etc. Nevertheless such adjustments are still not good enough to provide a context-aware and usable UI, because users do not share same profiles and preferences, interact via different devices, from distinct environments and use multiple modalities. Accordingly, context-awareness of user interface has been considered as a potentially important factor of their usability [Motti et al. 11]. A fair amount of research has been conducted to identify and support developing
adaptations in order to streamline interaction with systems. MICAA aims at providing UIs the ability to realize adaptation decision with regard to the ambient context while improving their acceptability among the end-users. It considers user interactions and feedbacks to refine the context-awareness of system adaptation decision, in order to increase the performance of context-aware adaptation at runtime.

An extensive range of technologies support the advancement of interaction context-awareness such as: Multi-Touch operation, pattern, gesture user recognition, facial expression recognition, natural language input, eye-tracking to follow the attention of the user and also body-gesture-based operation (Fig. 1)[Guellersen et al., 02, Wilson et al. 2010, Continental 2011]. Fig. 1 shows some UI exploring advanced technologies for different interaction scenario.

MICAA’s main purpose is to streamline the interaction and bridge the gap between user’s needs and interface customizations during execution (are runtime), which is rarely considered: current applications are not particularly intuitive, usually unclear and provide very conventional adaptations. The user is mostly considered as a passive receptor for adaptation and typically expected to be a trained specialist, which is not always the case with the widespread use of technology. Context-aware adaptation should consider runtime background and end-user preferences to clearly reduce interaction efforts as well as error rates.

Fig. 1. a). One single the swipe of a hand is enough in the Magic User Interface by Continental to move content from one screen to another [Continental 2011]. b. Users picking up objects from the table at a public demo of LightSpace Picking up objects from the table is accomplished by swiping them into one’s hand; following the pick-up, one can see an iconic representation of the object (a ball) in their hand [Wilson et al. 2010].
1.2 Thesis Statement

This research is aimed at enhancing the interaction quality by improving the UI context-awareness following a mixed-initiative adaptation approach. Considering user interventions within a MICAA support an effective interaction. It grants access to a straightforward users’ guidance through feedbacks during the interaction to reinforce adaptation context-awareness, besides the extraction of further supplied acquaintances for the UI customization. The major purpose of this thesis consists of investigating advanced algorithms to realize a user-centered context-aware adaptation during runtime. According to this perspective, performing the right adaptation during runtime requires the development of an advanced adaptation algorithms.

1.2.1 Research questions

Since current UI adaptations show limited end user involvement in specifying and governing adaptation, this thesis addresses the following major question:

*How can MICAA be leveraged for improving interaction with GUIs?*

To better illustrate and identify the research questions of user-centered scalable adaptations, we illustrate addressed requirements from both sides (users and UIs) through the loci argumentorum of Quintilian (who, what, how, where, when, with what) deriving from the main research question. This allowed “decomposing” and evaluating it in a “piece-wise” manner [Paramythis et al. 10].

Fittingly this question could be addressed from both sides of the interaction: user and system.

- **User**
  - **Who** is responsible for defining context-aware adaptation at runtime?
  - **What** individual user data could be considered for runtime adaptation?
  - **How** to bridge the gap between adaptation decisions and users preferences at runtime?
  - **When** to involve users in adaptation?
  - **Where the** runtime adaptation should be computed?
  - **With what** means adaptation could be endorsed to increase user satisfaction with interfaces whilst complying with contextual requirements?

- **System**
  - **Who** validate acquired knowledge and compute adaptation decisions?
Chapter 1. Introduction

- **Which** techniques to use to involve end user throughout the adaptation process?
- **How** to adapt the UI with regard to user preferences?
- **When** adaptation should be displayed for the end user?
- **Where** and at what level of intrusiveness user could be considered (user capability)?
- **With what** techniques the system could validate acquired knowledge and compute adequate context-aware adaptation decisions?

1.2.2 Thesis statement

In this thesis, we argue for the involvement of end users and operating advanced algorithm for deciding context-aware runtime adaptation would allow UIs an effective interaction enhancement improving UI context-awareness and user centeredness.

Hence, the central goal of this thesis is:

To define, to conceptualize, and to implement a mixed-initiative approach for context-aware adaptation (MICAA) of user interface of an information system at runtime based on user feedback; to apply this approach respectively to abstract UI identification, concrete UI specification, and content recommendation.

The main goal of this thesis consists of strengthening effectively mixed-initiative adaptation approach to support runtime context-awareness. This goal is achieved through a list of contributions supporting all MICAA development phase through different abstraction levels (Fig. 2).

- **C1: UFont.** An ontology for specifying feedback in (MICAA) mixed-initiative context-aware adaptation of graphical user interface.
- **C2: MICAA-MM.** meta-model for conceptualizing MICAA focused on C1.
- **C3: MICAA-Str.** A structural arrangement of MICAA according to six aspects: to what, when, why, what, where, how, who.
- **C4: MICAA-MF.** A methodological framework for executing MICAA in an agile paradigm based on C2 and C3.
- **C5: MICAA-IQS.** An interaction support structure for defining three properties for assessing MICAA quality.
• **C6**: Three C4 instantiations: for identifying abstract user interfaces, for selecting concrete user interfaces and for recommending contents.

• **C7**: AUIalg. A family of adaptation algorithms for identifying the abstract user interfaces.

• **C8**: CUIalg. A family of adaptation algorithms for selecting concrete user interfaces.

![Fig. 2 Thesis contributions](image)

Contribution this thesis are summarized by the following mains points:

• **C9**: CARRec. A family of adaptation algorithms for context-aware recommendations of news items.

• **C10**: M@rina. A java implementation for supporting C7 at runtime.

• **C11**: WISEL. A PIM based software for supporting C8 at runtime.

• **C12**: A series of experiments for evaluating (testing) C7 and C8 as implemented in C10 and C11.

### 1.2.3 Aims and Scope

The main intent of this research is to reply to the abovementioned research questions by developing a many-sided approach for supporting the development of adaptive enterprise application UIs using runtime models. Therefore, MICAA intended to streamline the interaction with nowadays
Chapter 1. Introduction

feature-rich application. Usually interfaces are complex and difficult to manage in the available heterogeneous contexts considering an individual user’s perspective.

One means for helping users cope with this complexity is to provide them with an interaction scenario and a Graphical User Interface (GUI) that is suited to their specific needs, skills and expectations. A customized interface that is tailored exclusively to a user’s needs and preferences could increase its usability level. Accordingly, involving users is now a prerequisite for any system wishing to maintain a competitive usability edge. In this respect, MICAA investigates a middle ground between two styles of adaptation: (1) the adaptable approach, where personalization is fully user controlled and fill his own preference, and (2) the adaptive approach, where personalization is fully system controlled and reduce required user workload. The MICAA adaptation approach contributed by this thesis is divided into the following main parts:

First we introduce a methodological framework for mixed initiative context-aware UI adaptation at runtime, which is generic enough to be applied in multiple approaches and different domains. We give a lot of importance to the user involvement through feedbacks during interaction. Besides, a particular attention for the agility of adaptation decision making characterized by determinants, constituents, ends and directives to support runtime context-awareness.

For the purpose of making clear the usefulness of the developed methodologies, we outline different adaptation scenarios and algorithms conforming MICAA. The instantiation is based on machine learning techniques and user feedbacks. It aims to provide an agile adaptation process, to support runtime context-awareness and also to deal with unexpected complex or fuzzy information. The proposed approach is applied to an illustrative case study of car rental interface involving user preferences during interaction by means of both explicit and implicit feedbacks. The system adapts itself by changing the arrangement of interaction units and/or redefining the whole scenario of interaction by predicting the user behavior. We assume that adjusting the scenario to user preferences and behaviors improves both usability and accessibility. Such adaptation could be initiated by both user and/or systems and executed at runtime.

1.3 Thesis Outline

This thesis is organized as follows (fig. 3):
Chapter 1. Introduction

- Chapter 1 introduces the context of this thesis, motivates and defines its central goal. It introduces related concepts, research questions and gives an overview of this thesis,
- Chapter 2 reports on a state of the art of works related to involved practices: Context-aware adaptation definition and methods, Techniques supporting mixed initiative context-aware adaptation, and the agile process for supporting context-aware UI adaptation. It provides main definitions and important shortcomings of mixed-initiative context-aware adaptation, besides discussing the benefits, risks and challenges of considering such support.
- Chapter 3 describes the computational framework created based on the identified shortcomings, to provide a common ground for understanding, discussing, analyzing comparing and even inventing next intelligent interaction style. This chapter gives extensive information on conceptual contribution related to MICAA from different viewpoints and discusses their cost benefits.

Chapter 4 reports the specification of the case studies; it lays down a MICAA instantiation considering the abstract UI. It starts by defining a list of algorithm applying ML techniques for AUI identification, a java...
Chapter 1. Introduction

implementation “M@RINA and an evaluation and analysis with regard to the applied theoretical framework presented in chapter 3;

• Chapter 5 lays down a second MICAA instantiation targeting the widget selection issue. It start by stating a list of ML algorithm for CUI selection, a web implementation “WISEL” and an evaluation; Chapter lay down a third MICAA instantiation contributing content recommendation issue.

• Chapter 6 depicts a third instantiation of MICAA addressing content adaptation. We discussed MICAA-based solution for improving news recommending. We established then some avenues of exploration of recommendation via JouNum project.

• Chapter 7 concludes this thesis by summarizing its contributions, discussing its shortcomings, and presenting future avenues to this work.
Chapter 2  Related Work

UI adaptation approaches need to be revisited with the hypothesis that they are going to be utilized by different users in countless and changing contexts of use. Accordingly, the critical requirement for efficiency must be handled continuously during execution with regard to ambient context. Nevertheless, up to now to select the appropriate adaptation during execution is still ambiguous and sometimes difficult to reach. Moreover, handling recurrent trade-offs. For instance, adaptability vs. user performance and adaptivity vs. user expectation is still challenging. In this regard, this chapter presents and discusses current research related to context-aware UI Customization at runtime.

Related works are initially explored based on their relevance to point out involved background information for context-aware adaptation. Such exploration leaded us to identify the main shortcomings that motivate and drive the requirements for this thesis. The works reported in this chapter were also necessary to realize the purpose of this thesis.

Section 2.1 defines adaptation and provides an extended analysis and categorization for general considerations and practices related to context-awareness. Section 2.2 reviews the main concept supporting MICAA. It presents the integration requirement of HCI and ML to improve context-awareness. Section 2.3 discuses MICAA process and highlights the need for agility for supporting aimed context-awareness. Section 2.4 illustrates MICAA’s crucial quality criteria with regard to the iso9241-110. The section is concluded with a set of key points backing MICAA.

2.1 Adaptation in HCI

In order to structure the state-of-the-art, this chapter presents selected works related to context-aware adaptation and then sheds some light on MICAA solutions for the development of context-awareness. We first investigate adaptation approaches, their functioning, practical complexity and truthfulness.
Interaction context-awareness becomes an underlying factor for supporting UIs’ quality and usability [Motti et al. 11]. It emphasizes the interface’s capability to fit changing context supplies and to improve the interaction quality and the users’ experience at runtime. Adaptation decision consists of a ‘needs’ assessment to identify the gap between the current interaction context and what user would like the interaction to be, further the execution of appropriate adaptation that meet situation requirements. Improving interaction capitalizes mainly on adaptation and flexibility:

*Interaction adaptation* denotes the capability of the UI to adapt to the situation or to be adapted considering the context constraints, where the context of use is understood as the combination of a user, intekkplatform and environment [Calvary et al, 03].

*Interaction flexibility* denotes the diversity of feasible exchanging modes of information. For instance, online flight reservation system may use graphics maps as well as text to provide users with comprehensible data.

Recent technological drawbacks provide new opportunities for adaptation practices at runtime improving context-awareness and overcoming existing approaches’ drawbacks, e.g. advanced sensors assessing contextual changes at runtime, advanced algorithms for acquiring knowledge and making decision at runtime [Motti et al. 11]. Such advances support the HCI community’ challenge for designing appropriate responsive adaptation at runtime.

Many techniques were defined and deployed, but there is still no agreed technique for learning, deciding and carrying the appropriate adaptations during interaction neither accurate approach to manage ambient unexpected circumstances. Despite the fact that, systems are needed to be flexible upgradable and self-learners over time considering runtime-acquired knowledge the automated adaptation remains as double-edged swords, since enhancements of UIs’ pervasiveness and proactivity could improve usability but also disturb users and interrupt their interaction.

### 2.1.1 Adaptation based on context information

Given the variety of contexts of use and user demands adapting UIs requires more complex inferences and strategies for considering up-to-date contextual facts. Although there were effective adaptive systems 10 years ago (e.g., [Knutov et al. 09, Findlater et al. 08, etc.]), they did not often make use of particular users’ preferences and context’s circumstances at runtime. A context-aware adaptation should have a crosscutting impact on software design and
appearence depending on the interaction’s context with an insignificant cost [Motti et al. 12, Totterdell et al. 90].

Due to the wide range of application domains, aspects and contexts of use, it is not scalable for human programmers to create UI versions for each scenario, instead an automated evolutive solution is necessary [Gajos et al. 04]. Most of the existing works assumes and supports the interaction from conventional context of use. However, this approach is not suitable for the current technological scenario. Mainly, because users do not share same profiles and preferences, interact via different devices, from distinct environments and use multiple modalities, context-awareness of user interface has been considered as a potentially important factor of their usability [Motti et al., 2013].

A fair amount of research has been conducted to identify and support developing forward-looking adaptations that meet unexpected situations and streamline interaction with systems. This perception is supported by latest circumstances:

- **From the technological perspective**, there is an increasing offer of new technologies and devices, whose characteristics vary, which provides new opportunities and requires new advanced adaptation practices. Device’s capabilities vary in terms of mobility (e.g. dimensions and battery life), network access (e.g. Bluetooth and wifi), quality (e.g. screen resolution and audio quality) sensors (e.g. geo-localization and sensors) etc.

- **From the end user’s perspective**, end users become more demanding, and interfaces have to maintain usability within the growing complexities of functionalities and the changing contexts. Users expect interfaces that are flexible, plastic and perfectly adjusts themselves according to the context of use in which they are located, i.e. optimizing the resources consumption, providing higher usability levels and greater user experiences. Further, system support and services are required to be in place to ensure that end-users are ‘properly’ considered, with regards to their own preferences and needs.

To achieve this outcome, ‘Context-awareness’ is the crucial requirement for an improved quality of interaction. In this sense, adapting user interfaces ‘intelligently’ in the executing environment in response to context changes, such as: location, resources, platform rather than satisfying user needs, improves the interaction quality. Intelligent adaptation refers to an advanced logic.
Chapter 2. Related Work

Several information about interaction’ situation need to be considered to better support context-awareness. Commonly, contextualization is regarded through mainly three adaptation' dimensions.

- The users’ profile in term of needs, wishes preferences, possible impairments and constraints.
- The interaction devices, (platform) which require their characteristics to be well-known and properly take into account in the adaptation.
- The environments, which provide information about situation of interaction. Devices sensors offer a significant awareness, examples of sensors of brightness, noise, localization and stability levels.

A. User

User is probably one of the most complex dimensions of the context since it continuously changes over time and involves a large amount of characteristics. Indeed

Koch (2000) describes the application of user models as follows: “Users are different: they have different background, different knowledge about a subject, different preferences, goals and interests. To individualize, personalize or customize actions a user model is needed that allows for selection of individualized responses to the user”. User modeling is an important research topic in its own right.

- User models capture a collection of personal data associated with a specific user, including cognitive skills, intellectual abilities, intentions, motor capability, and interactions preferences with the system. These properties could be assessed dynamically or statically depending on its possible variation over time. Common user models have been defined as a stereotype of a user group based on descriptors [Rich 89]. Users’ characteristics can be considered explicit depictions of the properties of an individual user and can be used to reason about the needs, preferences or future behavior of that user. User’s properties fall into three categories: cognitive, affective and conative [Cohen 90, Fischer 12, Rasmussen 86]. The creation of a User Profile (UP) for context-aware application focuses on the modeling of both dynamic and static user aspects. [Wahlster et al. 89] stress that a user model is a knowledge source, which is separable by the system from the rest of its knowledge and contains explicit assumptions about the user. Finin (89) argues that a user model is 'knowledge about the user, either explicitly or implicitly encoded, which is used by the system to improve the interaction. User profiles could be defined in several ways [Spyridonis et al. 14], the most frequently used being:
• **Model-based User profile:** Modeling user was the common interest of different project for user interface adaptation. It consists on defining a stereotype of common human attribute. Recent theoretical developments, as well as ongoing work on adaptation frameworks in various HCI fields, provide an opportunity for reviewing the current concept of user modeling. The development of adaptation projects has enabled a high level of standardization integrating all works e.g. (VERITAS, VICON, MyUI, GUIDE and VAALID) [Spyridonis et al. 14]. Their cooperation resulted into Virtual User Modeling and Simulation Standardization (VUMS), proposed common exchange format including a large set of variables describing various human characteristics (i.e., motor, visual, hearing, cognitive)[Kalkanis et al. 14]. The VUMLS profile is one of the most expressive user model used today [Spyridonis et al. 14].

• **Ontologies based user profile:** This approach allows to define a user description with semantic contents. Ontology has been a basis for building a user model in several personalized systems such as information delivery systems and Intelligent Tutoring Systems [Dicheva et al. 00] [Middleton et al. 04]. Several user aspects could be covered, for instance user preferences, user behavior, user personality, user need and actions etc. Tailoring interfaces (content or visualization) with regards to user profiles is called personalization. Similarly to adaptation, personalization could be achieved through two techniques automatic customization (adaptivity) and manual customization (adaptability), where automated personalization considers information about user during interaction from observing their behaviors and feedbacks. And manually personalized UIs system capitalizes on allowing user to modify the interface regarding his/her preferences through direct manipulations. Information about users can be gathered in several ways: asking for specific facts, learning users' preferences by observing and interpreting their interactions.

• **Highly dynamic user models:** This model allows a more up to date depiction of users. Interests and preferences changes are noticed progressively and influence the user models. Gathering live uses data pointed to the need for user involvement, which is typically needed in order to verify and/or rectify the result of UI adaptation and endorse system decisions [Kujala et al. 05]. An advanced adaptation has to consider pre-existing knowledge besides novel acquired data about users experience with the system. Such experience reflects users preferences and needs, which should to certain extent to move the adaptation engine by means of his/her interventions and feedback. Not only the users should be able to accept, reject or change adaptation rules, but also
Chapter 2. Related Work

the system must be able to learn from the users interventions for adjusting decisions, improving the system performance and accuracy.

- **User feedback:** Truthful users requirements could be identified only at real time during interaction. Accordingly user feedbacks should be beneficial means for understanding user needs and preferences. At its most basic levels, User Feedback (UF) is information provided by users about his preferences and/or goals. It gives information on the accuracy, adequacy, correctness and appropriateness of the system state.

The usefulness of UF depends on several factors such as the nature of feedback stimulus and how it is handled by the system. In order to establish an effective personalization, user preference needs to be learned [Shneiderman 97]. Accruing information about users based on their interaction and feedback remains faithful to real user preferences as long as it is correctly interpreted. As early as 1983, Ramaprasad [1983] defined feedback as “information about the gap between the actual level and the reference level of a system parameter which is utilized to alter the gap in some way”. Numerous studies deal with the end-users interventions in terms of implicit and explicit UF; such studies also investigate which data to gather and provide respective approaches.

| Table 1. Characteristics of explicit and implicit feedback for interactive systems |
|-----------------|-----------------|-----------------|
|                | Implicit feedback | Explicit feedback |
| Accuracy        | Low             | High            |
| Abundance       | High            | Low             |
| Context-sensitive | Yes            | Yes             |
| Expressivity of user preference | Positive | Positive and Negative |
| Measurement reference | Relative | Absolute |

As every user interaction can contribute to an understandable interest-indicator [Claypool 01], many researches give intention to the unconscious interaction as useful data for adaptation [Leiva 11], [Eisenstein 00]. However, as [Hu et al. 08] and [Jawaheer et al. 10] remark, the implicit feedback does not illustrate a dislike-attitude and outcomes a high inherent noise. As well knowledge can be acquired by asking the user during the interaction in an unambiguous way, which shows generally more expressivity than implicit feedback. By the same token, [Eisenstein et al. 00] recommend the use of satisfactory analysis techniques as the most accurate practice for adaptation decision-making.
Chapter 2. Related Work

The literature offers several techniques to gather explicit/implicit feedback [Claypool 01, Hu et al. 08, Eisenstein et al. 00]. Both explicit and implicit feedback provides different degrees of accuracy and expressivity besides varying degrees of investment and commitment to deliver the expected benefits. [Eisenstein et al. 00] compared both implicit and explicit feedback in term of accuracy, abundance, context-sensitivity, expressivity and measurement references through music recommendation systems. Results are presented in the table 1.

B. Platform

The platform dimension and their resources denote the device or set of devices holding the interaction. Several characteristics of both hardware and software aspects can be considered: screen dimensions and type, the battery level, processing capabilities, and the network availability and configuration. Several works considered adaptation to the platform. E.g. CC/PP (Composite Capabilities / Preference Profiles) [W3C 06] typically falls under this category.

The platform is described at four levels: the hardware defines the physical characteristics of the platform; these characteristics relate both to the input/output devices as computing capabilities of the platform. The software level this level relates to the operating system and installed software. The browser is in particular described by its name (for example, Mozilla, Internet Explorer, etc.); and the network level, defining the server on which the current session is open, the type of supported security (e.g., PPTP).

Several new interaction techniques generated by the sensors were reported by [Henkley et al. 00] for instance recording memos when the device is held like a cell phone, switching between portrait and landscape display modes by holding the device in the desired orientation, automatically powering up the device when the user picks it up the device to start using it etc.

C. Environment

The third context dimension denotes the socio-organizational environment in which the user is interacting. Although the relevance of considering such information to customize UIs, the adaptation regarding environment was imprecisely advanced. This inaccuracy is mainly due to the additional requirement for advanced resources and skills of the platform.

Environment was mainly cogitated through considering luminosity, noise levels and some social descriptors of the physical environment holding the interaction. Such dimension has mainly impact on two ergonomic criteria the readability (accessibility) and the ecological validity of interface in term of respecting social conveniences.
2.1.2 UI Adaptation: Granularity of changes

Different functional dimensions could be considered to identify and characterize adaptation [Lane 1990], for instance the execution time, UI portability and adaptability. Likewise the change granularity resulting from an adaptation behavior is an important criterion that can be used to predict the effects of an adaptation and associated cost/benefits. The change granularity may be required to identify the specifications that immediately precede the gross structural design phase [Lane 1990]. The granularity level of an adaptation has an important impact on the final implementation of UI and its usability. [Florins et al. 06] focused on graceful degradation of user interfaces. They proceed to splitting rules to transfer the interface between different platforms.

Furthermore we believe that defining the adaptation approaches need to consider the adaptation’ granularity levels. Motti [2013] defined the granularity considering the interface component. Three levels were identified: interactor level, dialog level and total level. [O’Donovan et al. 15] deals with adaptation granularity level from a different perspective. DesignScape provided users controllability on adaptation both for large-scale layout changes, as well as small improvements in position, scale, alignment, and line breaks. [Peissner et al, 13] distinguish adaptations depending on their costs and benefits as follows.

- Adaptations with high benefits help to overcome an individual interaction barrier.
- Adaptations with low benefits have no positive usability effect for the user in the current situation,
- Adaptations with high costs require an extra interaction effort compared to the user interface before adaptation,
- Adaptations with low costs do not require extra interaction effort [Peissner et al, 13].

An extensive design space and design rules for UI architecture is provided by [Lane 1990]. Lane considers adaptation granularity as Application portability across interaction styles dimension. What degree of portability across user interface styles is required for applications that will use the user interface software? [Lane 1990]. Three levels were identified:
• High: Applications should be portable across significantly different styles (e.g., command language versus menu-driven).
• Medium: Applications should be independent of minor stylistic variations (e.g., menu appearance).
• Low: User interface variability is not a concern, or application changes are acceptable when modifying the user interface.

With regards to existing classification of adaptation granularity, we established a synthesis of existing works. Four levels are recognized and described in order to better characterize the adaptation’s scope, capacity and influence. Established classification provides an overview of the potential UI changes through fous granularity levels. Identified levels ranges from global across the entire software specification down to simple layout refinement.

![Fig. 4 Adaptation granularity from Minor to Major changes.](image)

The low level considers simple adaptation, it consist on layout refinement with improve the interface style. At this levels adaptation are not functional and consider mainly the UI element style, for instance, simple
scaling of UI elements; rearrangement, e.g. changing the layout; simplification / magnification, same UI elements but with modified presentation;

The moderate level, at this level adaptation consists on a set of replacement rules. The replacement considers one or more UI component and does not result functional changes. For instance, for desktop UI version the hour selection through a long drop-down menu is a suitable choice while for a mobile version it is recommended to have a radio button with a limited number of options.

The high granularity level of adaptation, at this level adaptation support a deeper UI change as well as the UI expandability. For instance exploiting different interaction modalities (graphical, vocal, etc.) to achieve the same task.

The major adaptation represents extreme change of UI (i.e. versioning). This includes the changes to the overall design or system architecture that will be required to upgrade or adapt the interface. New function could be added to preexisting UI. Adaptations at this level are critical and may allow changing code. Fig. 4 depicts the fours levels through answering on <What? How? and Where?> questions

2.1.3 Approaches to UI Adaptation: a user involvement continuum

Assuming that it is a matter of priority for HCI researchers to develop a deeper understanding of adaptation trends and their implications for the users and interaction quality, we analyzed existing approaches and their impacts on the system usability and user satisfaction. A detailed depiction of existing adaptation approaches and frameworks is presented in Appendix A. We focus on establishing an adaptation continuum considering mainly the end-user. It serves as a calibrating tool for researcher to analyze, visualize and understand differences and similarities between existing approaches by recognizing opinions implicated for adaptation decision (user/system) and their implication degrees.

Interactivity and user involvement is an area of growing interest for designing both established and new systems. Since it is important to maintain consistent the user involvement and commitment for adaptation. On the other hand, automation of adaptation is supposed to lighten the interface handling workload and boosts user efficiency. Both visions are complementary, and when combined, create creative solutions with different considerations and with varying degrees of success. In what follow, we investigate different integration solutions and present techniques illustrating each solution. Different factor are considered for the analysis:

The adaptation determinants refer to the levels where adaptation is defined and the supported context of use <User, platform environment> and
Chapter 2. Related Work

the support of runtime. Adaptation could be supported at diverse abstraction levels:

- **Abstract level**: involves adaptation targeting abstract levels, it consists on customizing the arrangement of interface features and the interaction scenario regarding the context. For instance a minimal interaction scenario not including optional tasks should be displayed for novice user or on a small device.

- **Concrete level**: includes adaptation targeting the UI presentation and layout such as modality, interactors choices etc.

- **Content levels**: consists of adaptation that customizes content based on both usage semantics and context for instance predicting the appropriate interactive content and/or recommending content.

- **The adaptation concerns** refers to the deployed adaptation approach (adaptive, adaptable (see section 2.1.4). On the other hand it involves adaptation behavior supporting intelligent customization. The expandability of adaptation denotes the ability for enrichment of knowledge.

- **The expandability** allows the system to present a new adaptive behavior at runtime as needed to support a variety of aspects and factors.

- **Conflict management** enables a system to correctly manage adaptations in ambiguous cases by handling priorities. Such as adaptation rules priority.

- **The real time customization** denotes the UI’s capability to support on the fly adaptation and considers new changes during the same interaction session.

- **Runtime adaptation directive** relates to behavior intended to guide and support adaptation decisions at runtime.

  **Intelligent technique** denotes the system capability to intelligently infer knowledge, decides and recommends adaptation on interface and/or content. Evaluative feedbacks and direct manipulations refer to the user involvement with the adaptation decision process. We distinguish direct manipulation (user direct control) from evaluative explicit feedbacks that allow guiding the system control of adaptation.

Table 2. Identified criteria for the Characterization of adaptation approaches

<table>
<thead>
<tr>
<th>Approach for Adaptation</th>
<th>Adaptation level</th>
<th>Adaptation context</th>
<th>Adaptation concerns</th>
<th>Runtime Adaptation Directive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract UI</td>
<td>Concrete UI</td>
<td>UI content</td>
<td>Adaptation approach</td>
<td>Evaluative Feedback</td>
</tr>
<tr>
<td>User</td>
<td>Platform</td>
<td>Environment</td>
<td>Adaptable</td>
<td>Direct manipulation</td>
</tr>
<tr>
<td>Runtime support</td>
<td></td>
<td></td>
<td>Expandability</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Contact management</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Real time customization</td>
<td></td>
</tr>
</tbody>
</table>

35
Chapter 2. Related Work

Mainly five divisions of behaviors were considered with regards to the user involvement to adjust UIs to the context of interaction ranging from fully manual to fully automated adaptation. Several factors shape their practical behavior such as the level to which users can be involved in adaptation decisions and their participation ways (Control, collaboration, consultation).

Approaches must describe a definitive continuum of approaches, along which systems will fall (Fig. 5). In what follows we describe the established continuum drawing from existing approaches, through 5 stairs according to the degree of involving users, further we present some related tools for each category. Further, pros and cons are stated through positioning established existing works.

![Fig. 5 A continuum of user involvement for adaptation approaches](image)

**A. Fully-Manual customization: Adaptability**

Manual customizations are the fundamental approach to adapt UI. Adaptation decisions are defined at design time, where developer initiate systems with a set of adaptation rules regarding predefined ergonomic and usability guideline, and users are granted a controller role permitting to customize the interface presentation during use (adaptable).

A wide range of UIs is designed as purely adaptable systems considering the developer’s choices and preferences with regard to predefined guidelines and rules. Advanced adaptation can be performed during execution, whereas they still based on predefined and static rules. In this case, adaptation rules are manually defined and aimed to provide guidance for developers and/or end-users to customize the UI. Different works deployed various intelligent techniques to support adaptation decisions viewing a set of predefined developer knowledge’s. *Mastermind* [Székely 96] assumed an adaptable behavior for the definition of the concrete interaction object during runtime regarding environment (space) requirements. On the other hand, a large part of adaptable system capitalizes on user intervention to customize interface. Such applications are aimed to allow end-users to quickly, flexibly and seamlessly change the interface. For instance, *User interface Façades*
Chapter 2. Related Work

[Stuerzlinger et al. 06] enables end users to re-conFig. and re-combine GUIs by direct manipulation.

Rosson [1984] studied the user involvement with a text-editor to find out that programmers made extensive use of customization features, while secretaries did not. They also found out that the level of system experience was a reliable indicator of the customization done and consequently the user involvement. Mackay [Mackay 91] found out that users did not take advantage of the customization features as often or successfully as they could, and she concludes that frequent barriers to customizing software were lack of time and by the complexity of customization task. [Oppermann et al. 94] studied word-processing, spreadsheet, database, and drawing applications: a few users made UI adaptations and indicated that these options made their work easier and time-saving. However, many users reported that they simply made changes for curiosity or fun.

B. Mixed-Initiative Adaptation: MIA

The potential for mixed-initiative approaches is to strike the right balance between the adaptive and adaptable extremes. The optimal adaptation solution likely involves a combination of the two. In what follow, different mixed solutions are presented and classified with varying levels of user/system contributions on adaptation decisions

- Almost Manual customization

To overcome adaptivity shortcomings and improve the efficiency of the limited range of efforts customizing UIs the idea was to impart with the rest of the community and widespread best adaptation practices. Within this perspective, Crowd-sourced adaptation shows a potential to allow building an adaptive behavior drawn from individual users’ interventions to adapt UI.

Crowd-based adaptation: This approach is on the collective intelligence of a large set of users as opposed to a solipsistic approach of an isolated person. It involves a large set of end users to let best adaptation practices emerge. It capitalizes on users similarities at different interest level to generalize customization benefits. The key idea behind CrowdAdapt is to allow end-users to adapt the interface to their specific context, the current web page design do not support the interaction requirements. Drawing from individual user contributions allows the system to build an adaptive layout solution that caters for a wide variety of device characteristics and user preferences [Nebeling 13, Nebeling et al. 11, Akiki et al. 12, Lena et al. 16]. Nebeling [2011] extend RBUIS by supporting crowdsourced adaptation in order to speed up the adaptation process by engaging and leveraging the enterprise application communities. The approach is based on a context-aware toolkit that enhances existing web pages controllability, then users can adapt the UI directly in the browser. These adaptations are then deployed on a server and automatically downloaded and applied the next time that the web site is accessed from a
Chapter 2. Related Work

matching client [Nebeling et al. 11]. The approach is motivated by exploiting social connectedness, however privacy and security still need to be discussed in case of deploying approach in other context either than GUI. As well, both the quality of adaptations and performance of the system are not guaranteed.

• **Balanced Mixed-Initiative Customization**

User controlled adaptivity and conversational approach showed a typical collaboration between user and system through an elegant coupling of automated services with direct manipulation. Such approaches are centered on the relative promise of focusing user-interface research on developing new metaphors that enhance users’ abilities to directly manipulate objects versus targeting effort toward developing agents that provide adaptivity.

User-controlled Adaptivity: Other efforts focused on the convergence and harmonization of existing approaches. Fisher [2001] argued for the need for systems that mix adaptivity and adaptability to adjust UIs to changing environments and users. Such approach is intended to reinforce both context-awareness and user-centeredness. Accordingly, users are involved more explicitly in customizing the interface and adaptations decisions are made more accurately. Users interventions were boosted to a controllability’ prospect allowing the personalization of their UIs during execution. Examples of such actions or interventions include providing the user with assistance on how to use the application or automating part (or all) of the user’s task. In some case, the user is enabled to define directly the automated behavior of application. FlexExcel was the first system exemplifying a mixed initiative approach. Supple supported such explicit user intervention through feedbacks in order to modify the structure of the generated UI [Krogsaeter et al. 94].

Conversational-Adaptivity: In the case of conversational adaptation the user interaction capitalizes on the dialogue between users and interface: Where the user can propose his/her own training examples, however, the system can also directly ask the user for examples of certain tasks. Effort in discourse understanding falls into this category, where taking the initiative means guiding the flow of the dialogue.

Programming by Demonstration (PBD) is a technique for teaching the computer a new behavior by demonstrating actions on concrete examples (e.g. script and macro). This approach enables users to personalize their applications by automating repetitive tasks based on examples. The system records user actions and generalizes them into a reusable script for new examples. For example, SMARTedit [Wolfman et al. 01] is a conversational adaptive system, which learns repetitive text-editing procedures by example, MIMIC [Chu 00] relies on the cumulative effect of information dynamically extracted from user
utterances during dialogue interactions to provide more cooperative and satisfactory responses than existing non-adaptive systems.

- **Almost-Automatic Customization**

  Learning-based adaptivity considering Relevance feedback: Keeping the goal of personalization to improve the effectiveness of information access by adapting to individual users’ needs, feedbacks techniques was considered to support personalization instead of user direct control (manipulation). Eliciting feedback from the user to refine future interface decisions supports the trend of learning systems. The idea behind this approach is to create system that learns over time to be truly helpful and to build knowledge and develop skills while reducing training-related costs. Such approach offers an option to giving users indirect control of the adaptive mechanism, by extracting UF about the appropriateness of the decision, and considers the UF to improve future decisions. Several approaches were defined to support adaptation, such as learning from observation, and from knowledge’s.

  *Mondrian* [Lieberman 94] is an object-oriented graphical editor augmented with an instructible UI agent that learns procedures from examples. For learning based approach users play the role of a teacher, then the challenge is to make the computer reactive in its role as a learner. Adaptation aimed to be defined in a limited time with minimal user intervention (mostly via implicit feedbacks). *Supple++* [Gajos et al. 07] automatically generates UI tailored to an individual’s motor capabilities and that can be easily adjusted to accommodate varying vision capabilities.

  **C. Fully-Automatic Customization: Adaptivity**

  At the other end of continuum of (Fig. 5), adaptations are decided and managed by the system (fully automated). Such an adaptation provides for example an intelligent automatic interaction-object selection. Usability guidelines and knowledge are encapsulated within advanced techniques allowing system to infer knowledge and decides adaptation. However, such adaptation approach lacks predictability. Several works targeted ML techniques to enhance context-awareness. Among Knowledge-based approaches, Kleyn [1988] modeled a UI dialogue as an AND/OR graph, whose traversal results into possible sequences of actions a user might choose to invoke in accomplishing a particular task.

  *SURF* [Evers et al. 12] was one of first works addressing design and stylistics preference via adaptive algorithms considering end users: by solving the mapping problem by four main principles: Sensitivity, Understandability, Refinement and Focus such automation prevents designer intervention for the
concrete widget definition making the system difficult to adapt. Two other shortcomings were also observed: exceeding acceptable performance time [Nebeling et al. 11], insufficient relying on end users’ expertise for achieving usability [Akiki et al. 14]. Personalization and subjective knowledge denote a common sense theory of knowledge, however effective GUI design is not just common sense [Tullis 93]. Furthermore, no software developer can anticipate all user needs.

2.1.4 Discussion

Adaptation has achieved a great deal of improving interaction, but there is still much to be done. Several works analyzed and compared existing approaches from different perspectives. Adaptable and adaptive approaches vary depending on the amount of control provided to the user. Furthermore the time of making adaptation (design time and/or runtime) and considered context information contribute to the efficiency of adaptation and the interaction quality for end users. All approaches involve users in the design process, however the user involvement at runtime makes the distinction. Fig. 6 summarizes works analyzed above according to the proposed continuum.

An important difference between the various approaches lies in the degree to which users are able to govern the system design. For this purpose, ISO13407 [ISO1999] recommends the active involvement of users for understanding user and task requirements because they develop realistic expectations reducing resistance to change. The need for user involvement was largely investigated in HCI [Muller et al. 97, Kujala et al. 05]. [Damodaran 96] shows that effective involvement in system design yields the following benefits:

• Improved quality of the system arising from more accurate user requirements;
• Avoidance of costly system features that the user did not want or cannot use;
• An increased level of acceptance of the system;
• A better understanding of the systems by users resulting in more effective use
• An increased participation in decision-making within the organization.

Researchers in [Muller et al. 97] recognized three benefits for involving end user in participatory design: democracy, efficiency, and commitment. In the other hand, particularly regard should be given to circumstances such as the system performance. Accordingly, enhancing system decisions with regards to the end user needs could increase the system efficiency. In this sense, next
section expresses machine-learning techniques and provides an extensive investigation of its role in supporting HCI and adaptation.

![Diagram of adaptation continuum](image)

**Fig. 6 Existing works coverage on the adaptation continuum**

### 2.2 Support for MICAA

Several techniques and algorithms are supposed to be valuable for supporting MICAA approach. Particularly, advanced algorithms and intelligent techniques are able to contribute the MICAA. Such algorithms seem promising to enhance the adaptation decision-making process (adaptivity). Machine learning is an axis of the artificial intelligence field. It is one of the fastest growing technologies today with an abundance of prototype and application. The potential of the technology served different areas for instance robotics, computer vision, and manufacturing. The first definition dates back to 1959 by Arthur Samuel, who defined machine learning informally as the skill that gives computers the ability to learn without being explicitly programmed. ML techniques give the impression of being the most suitable for developing runtime context-aware adaptation. It is able to equip MICAA by appropriate algorithms to support knowledge acquirement and adaptation decision. In this section we study existing works that intertwine ML and HCI fields at different abstraction levels and we highlight benefits of ML in supporting the solution of HCI complex problems. A more detailed depiction of machine learning techniques and algorithm is reported in the Appendix B. The next section focus on providing a summary and for understanding the integration of both fields.

#### 2.2.1 Machine Learning techniques

In what follows, we provide a brief description of machine learning as an approach to gather a large collection of sample utterances from different people and learn to map these to world. ML solves different problems types that cannot be resolved by numerical methods alone. The goal of ML is never
Chapter 2. Related Work

to make “perfect” guesses, because ML deals in domains where there are extrapolations. The goal is to make guesses that are good enough to be a useful tool [Motti et al. 13].

Two main classes of ML techniques exist: supervised and unsupervised learning.

• Supervised learning refers to training programs. The algorithm learns from a predefined set of ‘training examples which facilitate the system ability to reach an accurate conclusion when given new data. Two major subcategories exist:
  - Regression ML systems: Systems where the value being predicted falls somewhere on a continuous spectrum.
  - Classification ML systems: Systems where we seek a yes-or-no prediction.

• Unsupervised learning system must find patterns and relationship among data to define a solution. There are no training instances used in this process. Instead, the system is given a data set and tasked with finding patterns and correlations therein. A fine example is identifying close-knit groups of friends in social network data.

<table>
<thead>
<tr>
<th>Table 3. ML techniques classification: Overview</th>
</tr>
</thead>
<tbody>
<tr>
<td>Techniques</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>Supervised learning</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Unsupervised learning</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Reinforcement learning</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
Table 3 summarizes the most used ML algorithms and provides some algorithm and problem sample for each technique. Moreover, more detailed description of ML algorithms can be consulted in the Appendixes B.

### 2.2.2 Techniques relevant for HCI

The evolving definitions of context-awareness and pervasiveness and interaction requirements increase the UI design complexity. To surmount such complexity, current researches’ interest focuses on advancing adaptation through intelligent techniques. Intelligent techniques are algorithms and method provided by artificial intelligence field to support automated reasoning.

The need for context-awareness challenges advanced algorithms and HCI to team up and fulfill the potential of both fields. Several works address this challenge by combining practices from user-centered design with the representation, reasoning, and recognition techniques that ML has to come up. HCI is urged to improve user satisfaction, so using ML techniques to recognize the user’s intent implicitly and/or explicitly, and decide meaningful adaptation can meet this requirement. Interface need to infer information about users based on some aspects of how interaction is managed.

Learning user preferences through interaction is intended to give HCI the opportunities to avoid frustration and reduce required workload. Although ML is able to provide several benefits for context-awareness, there is a lack of supporting tool and no agreed framework that aids developers in applying it. In this section we outline potential scenarios of context-aware adaptation where ML was successfully applied, presenting their common requirements and main trade-offs.

ML has been applied to support different façades of adaptation of human-computer interaction. Although these works are dedicated to exploring distinct applications of ML algorithms for context-aware adaptation, they are scattered, each one focusing on a specific application of adaptation at real time without a unified view of their potential benefits. Context-aware UIs invest more in user interactions and feedbacks. Accordingly systems need advanced algorithms for acquiring new information, to make conclusions and to acquire knowledge.

Although machine learning is able to provide important benefits for context-awareness, there is no practical guidance to developers in finding which approaches to consider for adaptation [Motti et al. 14]. CAA can benefit from ML potential especially during two distinct phases: the inferences about context information, and the CAA design decisions. Examples of ML application during the inference phase include: finding patterns in user interaction histories and clustering data, as users’ profiles [Jennings et al. 93]. Examples of ML applied for the design decisions of the CAA process include: predicting the user behavior [Mitrovic et al. 07], [Mitrovic et al. 09] automating
Chapter 2. Related Work

tasks, learning about the user preferences, and continuously evolving the adaptation engine itself.

Table 4. Roadmap of AI techniques for the support of HCI [Motti et al.2013]

<table>
<thead>
<tr>
<th>AI Technologies in interfaces.</th>
<th>Goals</th>
<th>ML Algorithms</th>
<th>Applicatiosns/Domains</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pattern-Recognition interfaces</td>
<td>The assignment of labels to a given input value, or instance, according to a specific algorithm.</td>
<td>Classification Segmentation Clustering Regression</td>
<td>-Speech recognition, Natural language</td>
</tr>
<tr>
<td>Knowledge-based interfaces</td>
<td>Making inferences from perceptual data, using tools from: statistics, probability, computational geometry, machine learning, signal processing, and algorithm design.</td>
<td>Classification Neural networks Decision tree Association rule</td>
<td>Medical &amp; Diagnostic Systems Industrial Control and Monitoring</td>
</tr>
<tr>
<td>Learning interfaces</td>
<td>Acquiring new, or modifying and reinforcing, existing knowledge, behaviors, skills, values, or preferences and may involve synthesizing different types of information.</td>
<td>Regression Decision tree and matrix Support vector machines</td>
<td>Robotics, Industrial Control and Monitoring</td>
</tr>
<tr>
<td>Intent-recognition / Prediction interfaces</td>
<td>Inferring user's knowledge and intention throughout interaction in order to arrive at a particular solution.</td>
<td>Segmentation Neural networks Markov chain</td>
<td>Financial &amp; Stock Market Monitoring and Prediction</td>
</tr>
<tr>
<td>Recording interfaces</td>
<td>Teaching the computer new behavior by demonstrating actions on concrete examples. Programing by example or &quot;programming by demonstration&quot;.</td>
<td>Neural networks Markov chain Dynamic decision tree</td>
<td>Intelligent Customer Support systems, Robotics</td>
</tr>
</tbody>
</table>

Along these lines, we provide a roadmap that guides stakeholders in the application AI technologies and precisely machine learning techniques for user-centered and/or context-aware adaptation. Potential application scenarios are exemplified, supporting the development of similar applications. We put forward a set of potential approaches that support the development of adaptive and adaptable applications in a challenging fashion.
2.2.3 Machine Learning Applied to HCI

We consider the assumption that any form of machine learning can be termed adaptation because the system acquires new information and reacts more appropriately to the new situation and novel circumstances [Norcio et al. 89]. Hence, there is an overall trend emerging toward the advancement of “intelligent” adaptation deploying more skillful methods.

**TRIDENT** [Bodart et al. 94] relied on a decision tree for selecting user interface elements depending on a domain model and an activity-chaining graph, a primitive form of a task model.

**TIMM** [Eisenstein et al. 00] expanded a decision tree by incorporating weights depending on the user model. The system employs the model-based approach by introducing abstract models that specifically target multi-platform UIs [Chu et al. 04]. As well, leading edge research in Machine Learning provides very accurate prediction tools, for instance,

**ADUS** [Mitrovic et al. 07] uses Markov-based models to predict user behavior and anticipate future user actions and to adapt the user interface accordingly [Mitrovic et al. 07]. In what follows we present a state of art on existing HCI works incorporating ML techniques for adaptation. We classify works regarding integration levels within UIs abstract specification. UI abstraction levels and the models presented below are detailed in Appendix A.

A. Machine Learning supporting AUI identification: Review

The Abstract User Interface (AUI) support a set of (sub-)tasks that are logically related from the user’s perspective in order to achieve a particular user’s goal, as well as the relationships between them. AUI has been developed for many different purposes according to the Cameleon Reference Framework (CRF) definition [Calvary et al. 03, Tran et al. 12, W3C]. W3C defines the AUI as a depiction of the UI in terms of interaction units without making any reference to implementation neither in terms of interaction modalities nor in terms of technological space. According to [Breiner 11], the goal of abstract UI is to supports high-quality access to the broadest possible range of targets for the broadest conceivable range of delivery contexts, via the broadest possible set of access mechanisms [W3C].

The majority of existing approaches for MBUID are in agreement that the AUI aims mainly to allow designers to create simply customizable graphical interfaces abstraction for various platforms, but does not describe the UI in a sufficiently abstract form to enable rendering in a final form [Calvary et al. 03].

Bearing in mind the above stated definitions and requirements of AUI, the development of operational systems requires the referencing of measurable criteria throughout all abstraction levels’ development. In spite of the context independent and defined requirements [Breiner et al. 11], several valid AUIs
could be generated for the same task model having diverse qualities and regarding different perspectives and quality levels. Dealing with a diversity of involved reflections over the AUI definition, we distinguish through features guiding the transformation process, assistance level, security and workload as main supported criteria.

Several works on UI adaptation focused on the abstract UI levels to define adaptation. Different contexts of use were considered, such as the platform for instance in Roam, Teresa and Maria approaches. Adaptation to the platform allows adding and deleting tasks with respect to the flexibility requirement of AUI. MOBI-D algorithm [Puerta et al. 97] is a good illustration of this approach, since the algorithm allows the definition of the different possible abstract user interface according to human factors. It aims at optimizing the structure of abstract containers and the decision support mechanisms. MOBI-D uses both user-task specification and domain models to make recommendations for presentation and interaction techniques however the algorithm was defined for different technological landscape and some updates are needed [Puerta et al. 97]. Both MARIA [Paterno et al. 09] and TERESA [Mori et al. 04] consider the same approach and addressed multidevice user interfaces by identifying relevant information to be contained in appropriate models, which allows for supporting migratory user interfaces.

![A task grouping sample](Mezhoudi 2013)

In the same axes SUPPLE, SUPPLE++, ARNAULD, and RBUIS argues for automatically generated user interfaces, which are adapted to a person's devices, tasks, preferences, and abilities, can improve people's satisfaction and performance compared to traditional manually designed interfaces [Gajos et al. 04, Gajos et al. 05, Gajos et al. 08, Peissner et al. 13]. SUPPLE, SUPPLE++, target mainly the automatic adaptation and elicitation of the parameters for the cost functions that guide the UI generation process [Gajos et al. 04, Gajos et al. 05, Gajos et al. 08]. Blumendorf [2010] approach allows for dynamically determining the (changing) environment situation requirements and (re-) distributing user interfaces at run-time. Most of existing
works targeted the generation of adapted UI at runtime. ROAM [Chu et al. 03] generates a prospected-AUI across different platforms in terms of three aspects of consistencies: task, layout and transformation consistency. The ADAPTS system [Brusilovsky et al. 02] include different models (tasks, users and environment) to adapt content and navigation in a hypermedia adaptable system. Although the variety of algorithms defined for the AUI definition, a lack of validity and consistency control still arises. Fig. 7 shows an example of potential AUI generated (Valid☺/Invalid☺) reified from a simple task tree. The focus of this point is to capitalize on learning based adaptation to improve the validity of the Abstract User Interface establishment.

B. Machine Learning supporting CUI selection: Review

“Concrete Interaction unit represents all visible, interactive object of a HCI, used for the information acquisition or information representation related to the user task in a specific context of use”[Vanderdonckt 95]. Concrete interface models (CIMs) specify a low-level description of UIs: A CIM is mapped directly to any GUI language implementing the interface. The CIM could be developed from the AIM or the Task model through a reification process performed by the designer or automatically.

The definition of the Concrete Interaction Unit CIU's in the concrete UI level is more associated with visibility purpose. It consists mainly on the selection of an appropriate widget for a task. The dependency between task type and widget is obvious, however many other factors should be taken into account when defining the concrete user interface such as; screen size, platform modality, user expertise etc. Context-aware definition of a widget can prevents usability, and ergonomic problems. For instance, the selection of appropriate widget in a graphical interface is intended to comply with visibility and accessibility guidance to guarantee the great usability level. The intelligent automatic selection of appropriate widget started back in 1995 with the first generation of model based user interfaces. TRIDENT [Vanderdonckt et al. 93] was the exception by considering a complex hierarchy and rules when generating the CUI model. Later [Szekely 96] adopted an adaptable behavior for the definition of the concrete element during runtime re-gradating space requirements, e.g. mastermind. [Puerta et al. 97] research was oriented to solve the mapping problem, which enables the design of a wide variety of types of UI. First researches assumed that a concise and well-defined set of rules, guidance and methods could put an end to the mapping problematic from abstract to concrete elements on an interface generation. Accordingly the requirement is to build systems that capture such knowledge sets and apply them automatically in the most efficient manner. However, such a precise set of rules is still infeasible and does not exist or that the rules are subject to so many variables. As the whole adaptation issue, the definition of concrete interface is a fuzzy process and includes subjective guidance that presents the user preference for instance. To our knowledge there is very little research that
has examined the adaptation issue through the CUI level, despite its prominence in Uls design and usability. Yet, until now, there has not been an adaptation approach considering widget and cogitating user during personalization. Based on that consideration, the identification of widgets could be seen as a recommendation issue. Given that a wide range of available interaction units could accommodate different interaction scenario’s requirements within different contexts of use and modalities, the identification of most appropriate widgets for tasks became a user reference purpose.

The CIU identification is intended to advance the process of creating a mapping between abstract specifications into a concrete one. At the concrete specification level, the mapping should consider the interface design rather the information design. It consists of planning the presentation of the interface to simplify and facilitate the understanding of the task purpose. Rather designing an interface element to facilitate user interaction with functionality. Therefore the improvement of this level have an effect on the visual aspect of the interface and influences the user experience during interaction.

![Fig. 8 Potential Multiple-choice widgets definition for a known domain.](image)

The literature conveys different approaches dealing with the automatic reification of concrete interface from abstract description, such tasks, and domain, to abstract user interface etc. Existing proposals were manual and automated, at runtime or design time and considers different contextual facts; TIMM [Puerta et al. 99] Bundelloni et al. 04, Dynamo-AID [Clercks et al. 08], MANNA [Eisenstein et al. 01] MASTERMIND [Szekely et al. 95].

The first generation focused mainly on the mapping of abstract interface definition into concrete UI. After that, approaches concentrated more on the adaptation and deployed more advanced technique to guide the
mapping. The intelligent automatic selection of appropriate widget started back in 93 with the first generation of model based user interfaces (Fig. 8).

TRIDENT [Vanderdonckt et al. 93] considered a hierarchy of object-oriented rules for identifying abstract interaction objects that were then turned into concrete interaction objects. Mastermind [Szekely 95] adopted an adaptable behavior for the definition of the concrete interaction object during runtime regarding environment (space) requirements. [Puerta et al. 99] researches were directed to solve the mapping problem, which enables the design of a wide variety of types of user interfaces.

COMET [Demeure 08] presents a software architecture style for building task-based plastic interactors. COMET brings together two approaches principles for plasticity: model-driven engineering and interactors toolkits. Interactors that are compliant to the COMET style are called COMETs. COMETs are expected to offer multi-rendering multi-technological, multimodal widgets. COMETs interactors are extensible and aimed to be controllable by the designers.

In order to perform the automatic mappings and interactor selection, a decision tree is used. Afterwards, [Mitrovic et al. 07] use Markov-based models to predict user behavior. ADUS [Mitrovic et al. 05] argue how the user behavior can be monitored at run-time in a transparent way and how learning methods can be applied to anticipate future user actions and to adapt the user interface accordingly [Mitrovic et al. 07].

C. Machine Learning supporting Content adaptation

Content adaptation is usually related to devices that require special handling because of their limited computational power, small screen size and constrained keyboard functionality. Context-aware recommender systems were evolving along with a rapid evolution of information systems and Internet. Methods such as collaborative filtering, content-based and knowledge-based recommendation gained huge popularity and recently most often in context-aware systems [Adomavicius et al. 11]. We present an investigation of existing recommendation techniques in Appendix D. The key rule of the recommendation methods is to exploit similarities measures. However, to perform meaningful content adaptation that best act for the needs of users and systems, it is required to know the characteristics of the whole context of use including possible adaptation that can be performed upon the content. Some of the methods for content adaptation are transcoding (changing the content fidelity); content layout re-arrangement; and distillation (extract the most important aspect). Existing works for content adaptation are developed to satisfy various purposes by different strategies. For instance, [Akiki et al.13] presented a role-based adaptation (RBUIS) allowing a content adaptation in GUI. It consists on simplifying the interface through two mechanisms: (1) by applying roles on task models and (2) by optimizing the layout by executing
adaptive behavior workflows (visual and code-based constructs) on concrete UI (CUI) models. The importance of contextual information has been recognized by researchers in many fields, including e-commerce personalization, information retrieval, ubiquitous and mobile computing, data mining, and management where content adaptation refers to context-aware recommendation. Such systems are aimed to generate relevant recommendations adapted to the situation of interaction. This topic is a relatively new research area having many underexplored and open research problems; some of them have been discussed in [Adomavicius et al. 11].

A. Discussion

Table 5. presents a comparison of the different adaptation techniques targeting different specification levels of interfaces: abstract, concrete and content. We analyzed works with regards to adaptation determinants, concerns and directives. We note that some criteria were not at all or not sufficiently addressed by any of the previous works. Most of existing approaches do not include mechanism for adapting UI to changing user preference and need during runtime. Users are mainly characterized by a predefined user profile such as in [Blumendorf 10]. Furthermore very little exploration work has been performed to involve users and allow them to explore possible alternative design throughout direct manipulation. Comet(s) should in concept allow end-users direct manipulation of UI design but this point was stated as future work. Aside from ADUS and ERICA most of technique do not offer any support for user preference during execution. The analyzed works did not support mainly two criteria: the expandability of adaptive behavior and the direct user involvement through explicit feedbacks. Accordingly significant improvement could be made on trade-off analysis, user satisfaction and system’s adaptation decisions. With the increasing mobility and pervasiveness of technology, software are more spread and used on or accessed by a variety people on several devices. This introduces significant and unpredictable dynamic variation in the context requirements and the users’ needs for the provided system. Expandability in terms of allowing system to infer, learn and decide novel adaptation with regards to new context is a requirement.

In order to give a qualitative description of the support of each criterion in described approaches we used Harvey Balls. They represent a user of round ideograms commonly used for visual communication. They are used in comparison tables to indicate the degree to which a particular item meets a particular criterion. Beyond the cost-savings benefit that visualization provides, visualization research over the past two decades has illuminated many other benefits. This visualization simplifies and organizes information by enabling perceptual inference operations and increasing the memory and processing resources available to the user [Kellen 03].
Table 5. Characterization of adaptation approaches based on [Roth 13] using Harvey balls (poor 🕗● ● ● Good support)

<table>
<thead>
<tr>
<th>Approaches for Adaptation</th>
<th>Adaptation level</th>
<th>Adaptation context</th>
<th>Runtime support</th>
<th>Adaptation concerns</th>
<th>Runtime Adaptation Directive</th>
<th>Evaluative Feedback</th>
<th>Direct manipulation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Abstract - III</td>
<td>Concrete - III</td>
<td>UI Content</td>
<td>User</td>
<td>Environment</td>
<td>Adaptation approach</td>
<td>Adaptation behaviour</td>
</tr>
<tr>
<td>Roam [04]</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
</tr>
<tr>
<td>TERESA [09]</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
</tr>
<tr>
<td>Supple [06]</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
</tr>
<tr>
<td>Arnauld [05]</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
</tr>
<tr>
<td>Supple+[07]</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
</tr>
<tr>
<td>Blumendorf [10]</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
</tr>
<tr>
<td>Crie do [14]</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
</tr>
<tr>
<td>REUIS [13]</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
</tr>
<tr>
<td>MYUI [12]</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
</tr>
<tr>
<td>Comet [08]</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
</tr>
<tr>
<td>ADUS [07]</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
</tr>
<tr>
<td>MANNA [01]</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
</tr>
<tr>
<td>Mastermind [95]</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
</tr>
<tr>
<td>Breiner [11]</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
</tr>
<tr>
<td>TIMM [99]</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
</tr>
<tr>
<td>Lum [02]</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
</tr>
<tr>
<td>CARs [11]</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
</tr>
<tr>
<td>TripAdvice [06]</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
</tr>
<tr>
<td>ADAPTNS [02]</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
</tr>
<tr>
<td>UNICA [06]</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
</tr>
<tr>
<td>MICAA</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
<td>● ● ●</td>
</tr>
</tbody>
</table>
Chapter 2. Related Work

The analysis of the related works and tools in the previous table, allows to identify that adaptation shortcomings is still the lack of systems that reinforce and support intelligent user centered adaptation. We derive three main requirements:

• The user direct involvement throughout direct manipulations. This requirement supports the UI controllability, which denotes the capacity and tolerance of system to support user-initiated customization of the interface. [Peissner et al. 03] argues for providing user full control over automatic adaptations in order to provide acceptable adaptive systems.

• The evaluative feedbacks that enhance the system predictability. Gajos [2006] considers that an adaptive system is predictable if it follows a strategy (logic) that users can easily model in their heads, and evaluated predictability effects on user satisfaction.

Advanced adaptation behaviors have to support expandability, conflict management and real time customization. We argue for advancing the practicability of involving intelligent technique toward an advanced adaptation. The goal is to make the UI more effective and efficient to better support the user’s current needs and changing context requirements while guaranteeing that the user can understand internal inferences. Still the main goal is to accommodate all constraints of the context of use and to improve UI usability and appropriateness to meet user increasing requirements and changing expectations. Nevertheless, there is still no efficient or effective adaptation solution that efficiently contextualizes UI's.

2.2.4 Discussion and final remarks

Conveyed adaptations in the literature are classified in three different modes: explicitly processed adaptation (adaptable), implicitly processed adaptation (adaptive) and a mixture of both [Opperman et al. 97]. Earlier, systems often needed recompilation for upgrades, which incurred increased cost, delay, and risk, however, in the current computational landscape of runtime, user centeredness, and context-awareness, the support of real time adaptation is required. ML as a domain capable of supporting the solution of complex problems is able to offer significant help [Alpaydin 04], [Bishop 06], [Barber 10]. Although these works are dedicated for exploring distinct applications of machine learning for UI design, computer supported cooperative work and adaptation.

As discussed above, ML techniques can provide a significant potential and convey the adaptation within UI in the executing environment. Through its several algorithms, ML seems suitable to manage numerous adaptation approaches and support systems in learning new adaptation rules with the main purpose of increasing the user satisfaction. The system should be proactive and
provide a correct reaction for corresponding events; those reactions should be learned and performed according to the end-user expectation.

Still, user preferences are the most relevant aspect to define adaptation, and human interventions are typically needed to verify and/or rectify the result of such adaptation. According [Rosman et al. 14], the customization of adaptation decisions is made more complex by the way in which users learn and the extent to which history can contribute in their choice behavior. These purposes require a more refined user model that supports the optimization process. To that end, different approaches have been proposed addressing adaptation problems; and introduced the context information at different levels. Almost all of them stimulate adaptation via an adaptive behavior [Blumendorf et al. 10][Bodart et al. 95][Breiner et al. 09][Breiner et al. 11][Chu et al. 04][Clerks et al. 04][Criado et al. 12][Mitrovic et al. 05][Mitrovic et al. 07][Paterno et al. 09][Eisenstein et al. 00]. Their primary goal is to ensure pervasive properties for user interfaces, and also having the ability to change the interactive application during runtime due to a contextual change.

Different adaptation purposes and context features have already been addressed. For instance, in Roam [Chu et al. 04], the authors apply models at runtime to build multi-platform adaptation. Then applications can run on heterogeneous devices and allow a user to move/migrate a running application among heterogeneous devices in an effortless manner. Adaptation in [Clerks et al. 04][Mitrovic et al. 05][Mitrovic et al. 05] considered user preferences, and such approaches supported the creation of context-sensitive user interfaces. TRIDENT [Bodart et al. 95], Roam [Chu et al. 04], DynaMo-AID [Clerks et al. 04], ADUS [Mitrovic et al. 05; Mitrovic et al. 07], Teresa [Paterno et al. 09], Breiner [Breiner et al. 09; Breiner et al. 11], [Criado et al. 12] derive a context-sensitive UI from task-oriented languages which allow for greater flexibility in generating user interfaces from the abstract specification. Commonly, existing systems provided some mechanisms to automatically generate user interfaces and used a simple rule-based approach, where each type of data was precisely matched with one type of interactor.

Several works have been dedicated to investigate ML techniques for HCI. However, normally they are constrained, focusing on deeply investigating: specific technics in a specific context and situations, or application domains. In this sense, the main shortcoming is that the works are limited, inflexible and not integrated, thus they do not consider a broad view of ML support for HCI.

Throughout several techniques, ML puts forward its ability to provide significant benefits for UI contextualization and convey the adaptation within UI in the executing environment [Alpaydin 04][Barber et al. 10][Bishop 10][Mezhoudi 13][Motti et al. 12]. Works that effectively involved ML techniques in finding the beneficial adaptation methods to solve contextual issues outline several benefits. Based on the fact that both machine learning
Chapter 2. Related Work

and human computer interfaces are cognitive sciences, and both focus on human and draw for human centered needs ML and HCI are converging. The marriage of both fields creates a unique relationship between content and container, which cross-fertilizes each other. Significant goals include, but are not limited to:

- **UIs Must Adapt Themselves to the Users**, Not the Inverse. An efficient implementation of adaptation, which considers changing user preferences and takes several contexts into account at runtime, is still the fundamental purpose for user interfaces. As a general rule, ML techniques process training input and offer support for decision questions based on this input [Lieberman 09].

- **UIs Should Assist Users in Achieving their Goals.** People are sometimes unable to access an appropriate application due to the complexity and depth of the menu structure. In this sense, user assistance provides information to help a person to interact with software. This can include describing the user interface, but also focuses on how to help the user to best apply the software capabilities to their needs. User assistance can be considered a component of the broader category of user experience. Effective user assistance development requires a variety of communication skills. These include writing, editing, task analysis, and UF. Since the user assistance is directly involved with software development, the discipline often requires an understanding of UI design, usability testing, localization, testing, quality assurance, instructional design, scripting or programming, and accessibility.

- **UIs must be designed to fit the context and the user requirements.** Nowadays technology offers a wide range of multifunctional devices that provide us with various useful applications and services anytime and anywhere. Adaptation is essential to solve problems with non-stationary spaces. However, this does not imply that prediction should be neglected, as it is useful and desirable. Nevertheless, prediction should be complemented with adaptation and used with caution, considering its limits carefully. If there are interactions, there will be a certain degree of unpredictability. This demands modesty and consideration when building systems and solving complex problems. The implication is that there will always be novel problems. The best that interfaces can do is to be prepared and expect the unexpected.

- **UIs Should Enable Cooperation.** In today’s UI, the division of labor between humans and technology is quite clear: Huge, automated applications are typically cordoned off in different devices and performing repetitive tasks, while humans interact in more hazardous environment on various goals requiring greater interaction feature [Lieberman 09]. During an interaction session user and interface undergoing an initial training session. Once the UI learns a person’s preferences, need and interaction
Chapter 2. Related Work

habits, it can be scheduled to recognize that same person, and initialize the appropriate task plan. ML can significantly improve the UI’s understanding of what the person’s next likely actions are [Langley 97].

• **UIs must perform interactions in order to make effective user inputs the user.** Besides usability testing in the initial development phase, such improvements can now also be based on the intuitions gained from real users by analyzing their usage behavior. The importance of monitoring global-level user interaction characteristics to adapt system requirements was proved in several works [Mitrovic et al. 07]. Based on captured user actions, the interaction graph enables proximity measurements across users, activities, resources, and artifacts. Besides continuous interaction monitoring and analysis is quintessential to enable software services to evolve in line with the overall ensemble structure.

• **Interfaces should make use of output and input methods that are natural to human users.** The interactivity of the system should be adaptable to the user’s requirements in given situations. The user should be able to choose between interaction that is mainly under the computer’s control and interaction that is under human control, and between an obvious interface that requires the user’s responsiveness and a transparent interface that is integrated in the experience of use [Lieberman 09]. By enabling the flexibility to choose the interaction model most appropriate for their understanding, motivation, and circumstances the interface control the trade-off between the user’s workload and his situation awareness.

**Potential trade-offs.** HCI has a challenge to fill the gap between what it promises and what it can reasonably deliver. These abilities hold out the assurance of easy-to-use, well-designed systems that meet real user needs and increased usability level. Although ML algorithms provide a set of benefits for context-aware adaptation, care must be taken in order to not disturb the user interaction. Almost as much as there are advantages when HCI and ML team up to fulfil the promises of both fields; there are several trade-offs that can be expected by automating UI changes by means of adaptation. Therefore if the user interaction is not very well understood, the adaptation results can: bother, annoy, confuse or even prevent users from achieving their actual goals [Lavie *et al.* 10]. That is why users must be always able to evaluate the adaptation, to confirm that the results achieved are convenient according to their goals and interests during interaction. Moreover, by changing the UI layout, that the users are familiarized with, can make them lost, thus all the changes must be clearly indicated and agreed by end user.

This section presented a literature review for ML as a technique supporting adaptation approach and their crosses in previous works. The analysis of literature review permitted the identification of the main shortcomings of UI adaptation and consequently the definition of correspondent requirements. The common requirements that are presented
Chapter 2. Related Work

below aim at fulfilling context-aware adaptation goals with ML, but also avoiding potential trade-offs.

2.3 Agile practices meeting adaptation requirement

Adaptation can be viewed as a short term or long term process or as an adaptive system in which different parts interact in order to obtain operational and usability. The current adaptation process challenge is to ensure a better understanding of context data to provide a meaningful guidance for the UI adaptation at runtime [Lim et al. 09].

Although there were successful adaptive systems 10 years ago, they did not often consider varying context information during execution. Given the changing status of user needs and expectations, adapting UIs often demands complex inferences and strategies for acquiring and considering up-to-date contextual facts. Likewise, adaptation should have a crosscutting impact on the software design and appearance depending on interaction features and the ambient-context with an insignificant cost [Totterdell et al. 90].

By attempting to cut with earlier interfaces that often needed recompilation for upgrades, which incurred increased cost, delay, and risk, UIs shift to a runtime adaptation paradigm. User interfaces turn out to be adaptive rather than being user-centered and carry out adaptation in accordance with the end-user preferences and context of use. Hence, a responsive adaptation at runtime is still a challenge in the HCI field since there is no agreed technique for learning and executing the greatest adaptation rules in case of unanticipated situations during interaction. Thus, interfaces need to be flexible and upgradeable over time considering contextual data accrued during interaction sessions, for instance the user's satisfaction levels. We focus on the high-level scope of context-awareness and UI proactivity considering an agile paradigm to enhance the UI context-awareness at runtime.

In this section, we demonstrate that to some extent the process controls and expresses the composition and the ordering of adaptation, defines couplings and establishes the major modifications in the system. We show the process outlines an approach and gives more understanding into its development and deployment. We examine the issue of aligning the adaptation process, based on a main requirement of responding to contextual changes efficiently and effectively ensuring a quick and agile reactivity. Adaptation process is intended to shift for a proactive phase, which decides changes, anticipates difficulties and take steps to overcome them, while being guided by users preferences. We outline the need for agility at runtime for context-aware adaptation to maintain UI proactivity.

Both adaptation and agile methods share the values of user-focus and iterative inspect-and-adapt cycles. However, there is still a lack of interchange
and integration between the different operated methods and disciplines. Similarities can be found in basic principles and practices as well as among the methods and tools that are typically applied [Noor et al. 08]. In spite of this, there are still many challenges that must be overcome in order to support adaptation at runtime with regards to agile principles. The aims are for:

1. Establish the different units that relay to the whole adaptation practices and to support diverse descriptions, implications and considerations of responsiveness.
2. To describe the lived experience of adaptation and evolution during execution,
3. To provide an agile progressive enhancement of runtime adaptation by ensuring the integration of different practices and techniques at runtime.
4. To support the new urge of researches on intelligent UI to systematically outline (intelligent/proactive/agile) context-awareness based on the advantageous idea of decomposing adaptation for evaluation purpose [Totterdell et al. 90].

2.3.1 Agile methods: Overview

In the last decade or so, the trend to more responsive development methods has been the most significant paradigm change after the use of the waterfall model in the 1970s. Since the agile manifesto creation in 2001, agile software development has attracted considerable attention of the research community. The term “Agile” refers to a software development methodology, which promotes a project environment of adaptation, teamwork, self-organization, rapid delivery and client focus. Despite some debated quarters [Turk et al. 02], the advantages of agility, including faster development, improved responsiveness to changes of clients requirements, and developed application quality, are incontestable to those who have mastered these practices [Allison 15]. Ranging from Extreme Programming (XP) through the methods of Scrum, DSDM, Feature-Driven Development (FDD), and Lean Software Development (LSD) to the iterative and incremental methods supported by the Rational Unified Process (RUP) the basic principles of agility have been effectively applied in thousands of projects [Irum et al. 14, Dingsøyr et al. 12].

Scrum and XP are the most used agile methodologies [Dingsøyr et al. 12]. Scrum is aimed at providing an agile approach for managing software projects while increasing the probability of successful development of software, whereas XP focuses more on the project level activities of implementing software [Allison 15]. Both approaches, however, embody the central principles of agile software development. Furthermore, a number of supportive agile practices have been applied within development methods and as stand-alone practices as well.
Agile approaches have gained a lot of attention in the software development field. It is defined as a methodology for the creative process that anticipates the need for flexibility and applies a level of pragmatism into the delivery of the finished product [Jameson 05, Andersson et al. 13]. In software engineering, agility refers to the viewpoint supporting mainly the capability for quick adjustment to changes in addition to the end-user involvement revealed at design time. A survey on agile method [Vijayasarathy et al. 08] specified that agile methods are mostly used for Internet, back-end and front-end development project (Fig. 9).

Fig. 9 Project types supported by agile processes [Vijayasarathy et al. 2008].

“These results suggest that while agile development is not confined to a particular type of software project, its inherent flexibility and responsiveness may be best suited for application that face rapid changes in both requirements and the facilitating technologies” (Fig. 9).

Table 6. Definition of 12 Agile principal [Beck et al. 2001]

<table>
<thead>
<tr>
<th>ID</th>
<th>Principle</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>The priority is customer satisfaction through rapid and continuous delivery of software that adds value to the business.</td>
</tr>
<tr>
<td>P2</td>
<td>Changes are welcome, even late in development, especially if the changes will provide a competitive advantage to our customers.</td>
</tr>
<tr>
<td>P3</td>
<td>Make frequent deliveries of software that works from a couple of weeks to a couple of months, always looking for the shortest time between deliveries.</td>
</tr>
<tr>
<td>P4</td>
<td>Business people (executives) and developers must work together daily and throughout the project.</td>
</tr>
<tr>
<td>P5</td>
<td>Build project around motivated individuals. Provide all necessary support to the project environment and rely fully on the team.</td>
</tr>
<tr>
<td>P6</td>
<td>Face-to-face dialogue is the most efficient and effective way to communicate the information within the development team.</td>
</tr>
<tr>
<td>P7</td>
<td>Software that works is the principal measure of progress.</td>
</tr>
<tr>
<td>P8</td>
<td>Agile processes promote sustainable development. The promoters, developers and users should be able to maintain a steady work pace indefinitely.</td>
</tr>
<tr>
<td>P9</td>
<td>The continuous attention to technical quality and good design enhances agility.</td>
</tr>
<tr>
<td>P10</td>
<td>Simplicity is essential. We need to know how to maximize work that should NOT be done.</td>
</tr>
<tr>
<td>P11</td>
<td>The best architectures, requirements and designs emerge from the team itself through its proactive and self-organization (collective and collaborative intelligence).</td>
</tr>
<tr>
<td>P12</td>
<td>At regular intervals, the team should reflect about how to become more efficient and adjust their behavior to achieve this goal.</td>
</tr>
</tbody>
</table>
Chapter 2. Related Work

Different researchers compare traditional and agile approaches, in their different perspectives such as development life cycle: development life cycle, architecture, requirements, team organization, style of development etc [Nerur et al. 05, Boehm et al. 04, Bohen et al. 05, Dyba et al.09, Highsmith 01]. Table 7 summarizes some selected differences between agile and classical development methods and life cycles.

Table 7. Differences between agile development and classical development methods and life cycles

<table>
<thead>
<tr>
<th>Perspective</th>
<th>Traditional Approach</th>
<th>Agile Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>Development life cycle</td>
<td>Linear, lifecycle model (waterfall, spiral)</td>
<td>Iterative, evolutionary</td>
</tr>
<tr>
<td>Style of development</td>
<td>Anticipatory</td>
<td>Adaptive</td>
</tr>
<tr>
<td>Goal</td>
<td>Predictability and optimization</td>
<td>Adaptation</td>
</tr>
<tr>
<td>Client involvement</td>
<td>Low involvement, Passive</td>
<td>User centered, Active/proactive</td>
</tr>
<tr>
<td>Management</td>
<td>Process-centric, command and control</td>
<td>People centric, collaboration</td>
</tr>
</tbody>
</table>

2.3.2 Adaptation requirement

Based on a common definition of agility, several HCI’s works [Noor et al. 08, Obendorf et al.08, Patton et al. 02, Silva et al. 12] advanced agile method for UI development. Commonly investigation was aimed at bridging the gap between both disciplines HCI and SE. A significant overlap was identified, such as in iterative design, small releases and prototyping, scenarios, testing and evaluation. [Patton 02, Silva et al. 12] demonstrate the contribution of agile paradigm for providing a beneficial support for HCI improvement within a user-centered paradigm. Commonly integration focused advancing UI development phase. On the other hand, tailoring adaptation for users preferences is still the key factor for the improvement of UI usability [Terada et al. 04]. Mostly, adaptations are performed when systems detect a context variation, by executing a particular reaction already encoded at design time. However, we argue that a successful Context-Aware Adaptation (CAA) [Motti et al. 12] needs to be more proactive and more user-centered by meditating new accrued data during interaction in an incremental way.

The adaptation process needs to be reexamined for both design time and runtime concerns. Agility is required during the deployment phase as well as for the development one. Two main concepts are required for adaptation at runtime: first the user-centeredness, and second an incremental and iterative enhancement. Both concepts match main agile practices.
Chapter 2. Related Work

Several analyses and studies targeted the development of adaptive systems from different points of view; most of them focused on the dimensions of adaptation and were specific for distinctive domains [Knutov et al. 11, Lavie et al. 10, Lim et al. 09] (medical, hypermedia, etc.). For instance, [Knutov et al. 09] proposed a classification for adaptive hypermedia methods and techniques by highlighting the adaptation process. Likewise [Knutov et al. 11] proposed a framework for categorizing UI adaptation based on two technical descriptions of two AUI key elements: the taxonomy of adaptation and taxonomy of triggers. Motti [Motti 11] proposed a generic framework for facilitating the development of context-aware application. The frameworks consist on two main parts: the context-aware design space and the context-aware reference framework. The most commonly cited issues with adaptive UI are the lack of flexibility, predictability, control, and privacy [Froelich et al. 09], mainly because those UI adaptations consider prior interaction knowledge (explicit context, domain models) [Chang et al. 09, Findlater et al. 08].

We are interested to extend the flexibility and provide system with the ability to learn and build novel knowledge in an incremental way in view of context changes. We focused on investigating runtime context-aware adaptation in depth to identify key factors for analyzing the adaptation during deployment phase. The idea behind adaptivity migrates from effortless flexibility into an intelligent responsiveness. Adaptations are expected to evolve continuously in a responsive and upgradeable way. Accordingly adaptation decisions should be determined throughout the system’s lifecycle from early stages in term of guidelines and predefined adaptation, intended to inspire the system adaptation engine, until the execution phase when system is required to be scalable and flexible. Thus, in the current computational landscape, the support of intelligent runtime adaptation becomes a crucial requirement, which calls for a context-aware agile adaptation. These built-in agile practices and adaptation skills, should lead to a significant assimilation of the increased complexity. In this sense, we identified related challenges and proposed a theoretical framework supporting agile context-awareness.

MICAA is proposed as a means to make systems customized or personalized with regard to end user preferences at runtime, thereby increasing the systems flexibility and usability. MICAA adaptation approach incorporating ML techniques and adopting an agile process follow intelligent UIs and shares aims to improve the efficiency, effectiveness and naturalness of human-machine interaction by representing, reasoning and acting on models of users, domain, task, discourse and media [Maybury 98]. To that end, UIs must be data driven, self-aware and have the capability to learn over time from the users experience.

Adaptivity is aimed to improve interaction, facilitate user’s tasks, reduce system complexity and avoid overloading users [Lavie et al. 10, Hook 99]. However, those benefits are rarely likely to be realized at the time the
adaptation decision is made automatically. UIs are almost inherently inconsistent over time i.e., their interface or functionality may change which may disturb interaction and augment the interaction complexity. Demonstrating that adaptive behavior improves interaction without violating usability criteria is still a challenge. An evaluation of adaptive behavior costs and benefits is reported in [Lavie et al. 10]. The impact of adaptivity manifests itself in terms of reduced effectiveness, which is in contradiction with the IUI basic features discussed in [Rissland et al. 84].

The main goal of MICAA approach is to find a way to maintain recognized benefits, and avoid additional development costs or disadvantages. MICAA promises to improve interaction’s quality for end users in term of: efficiency: allowing to achieve interaction goal with reduced workload. effectiveness: assuring the right adaptation at the right context, and neutrality: supporting the fluency of interaction considering end-user preferences. The validity of such improvement is approved by end-users, which might explain the growing trend of including UF at runtime in the adaptation process.

2.3.3 Discussion

A few years ago, the challenge of integrating agile methods with HCI was underestimated due to their differences in focus. However, nowadays there is a reasonable number of studies addressing this integration at design time, as can be seen in [Memmel et al. 07, Silva et al. 12, Noor et al. 08, Aoyama 98]. Both flexibility and agility are required to improve the UI adaptation. Each agility practice was widely discussed and showed a potential in different fields, for instance Software Engineering and User-Centred design.

We believe that the agile paradigm provides a beneficial support for HCI responsiveness at runtime as well as it was for design time. The goal is to reduce the gap between SE and HCI and consequently take advantages of agile practices to advance adaptation shortcoming at runtime. The main HCI requirement is still to improve the interaction and usability of the interface; which is valid and shared for different adaptation implementations. We examined agile practices and runtime/design time adaptations in terms of definitions, objectives and beliefs within their proper area, in order to underline fundamental concepts from diverse scopes. Based on such analysis we expand the reflection of intertwining agile practice with HCI. We focused on agile principles for UI adaptation by highlighting commonalities for both runtime and design time adaptation and then we overlapped their vision through an advanced UI context-awareness.

The context of use evolves over time; so adjusting the UI to comply with new requirements proactively should be expected. Thereby ‘Context-awareness” as well as ‘user-centeredness’ become crucial to improve the quality of interaction. We acknowledge that tailoring relevant aspects and practices of agile paradigm and reproducing them for the UI context-awareness at a
Chapter 2. Related Work

runtime should show potential for improving adaptation proactivity. [Patton 02] argues for the relevance of agile methods to improve systems usability defined as the extent to which a system can be used effectively and efficiently while satisfying a specified user.

Some review of relevant works supporting agile practices for HCI development were conducted in [Oppermann et al. 97, Seffah et al. 05, Silva et al. 12, Andersson et al. 13]. The significance of human-centeredness (HC) requirements to characterize agile methods [Silva et al. 12] provided a starting point for reasonable assumptions about the effectiveness of agility in the HCI field. From an agile perspective, user requirements are particularly prone to change and evolution, as the software application evolves.

This appears to address an important issue in HCI [Silva et al. 12] and can provide great benefits at design time, however this requirement is continuing and exceeds the design level, which make it more worthwhile to enhance adaptation during execution. Furthermore agile approaches often emphasize iterations as a requirement for the improvement of the software. As a result of improved iterative process and quick feedback, agile methods demonstrated its ability to support the successful software development. Similarly iterative process was recognized as the main design requirement to improve usability [Nielsen 93]. Such iterative development of user interfaces involves steady refinement of the UI features based on user testing, while the new trends of pervasiveness, and being iterative must be propagated and elaborated within the adaptation strategy in view of changing contexts of use and user preferences during interaction sessions.

Incremental paradigm would be another important aspect of agile approaches supporting a better knowledge transfer due to better user-system communication and frequent feedbacks from each iteration. Once again, the idea of incremental interfaces already exists, it consist on gradually increasing the UI complexity for a novice user by enabling advanced interface features incrementally as soon as the user needs and can use them. Such interfaces were developed for two intelligent learning environments: ITEM/IP [Brusilovsky et al. 95] and ELM-PE [Brusilovsky et al. 96]. Whereas this incremental aspect can be expanded to consider more context factors for instance the platform of interaction or the time. Further, incremental systems were based on predefined and static adaptation rules. Thus, they do not support the adjustment of UI complexity.

To extend the consideration of the above-mentioned agility practices to a more practical perspective, all should be considered at runtime and established within the adaptation process. In this sense, agile methods are able to address major outlined UI responsiveness shortcomings, like considering individuals, their interaction preferences and changing context of use, besides emphasizing the importance of human factors for adaptation during interaction.
Table 8 Similarities between Agile Method basis and HCI Practices

<table>
<thead>
<tr>
<th>Agile Methods basis</th>
<th>Description</th>
<th>HCI Design practices</th>
<th>HCI runtime’ practices</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature Driven</td>
<td>The system is segmented into sets of client-valued functionality, and development work is organized around producing these features</td>
<td>Modeling tasks, Scenario</td>
<td>Modeling adaptation rules, Context models, Context tracking, Decision models, knowledge models</td>
</tr>
<tr>
<td>Iterative, incremental</td>
<td>Development is performed in repeated cycles (iterative) and in portions at a time (incremental)</td>
<td>Prototyping, user tests</td>
<td>Contexts evolution, Runtime adaptation, user tests, Prototyping UIs, Learning Knowledge</td>
</tr>
<tr>
<td>Customer involvement</td>
<td>The Customer Involvement means accepting changing requirements and including the user and/or customer feedbacks in the development</td>
<td>User test, User-centered design, user experience</td>
<td>User involvement, User centeredness, user implicit and explicit feedbacks, User commitments, personalization, controllable adaptability</td>
</tr>
<tr>
<td>Team Dynamics</td>
<td>The collection of “soft factors” and effects related to unique practices that influence the development team’s performance</td>
<td>Design rooms, styles guides, collaborative design</td>
<td>Mixed-initiative adaptations, predictions, user controllable adaptability, System learnability</td>
</tr>
<tr>
<td>Continuous Integration</td>
<td>Continuous Integration involves methods of maintaining updated software</td>
<td>Evaluation, Usability Inspections</td>
<td>Adaptability support, controllability, Iterative prototyping</td>
</tr>
</tbody>
</table>

A number of interdisciplinary interfaces during the different phases of SE and HCI developments/deployment could be considered, for instance: iterative modeling, evaluation, etc. The pervasiveness and the responsiveness of systems over heterogeneous contexts of use became the common significant requirement of both fields. Accordingly, HCI and Agile Software Engineering can converge into new shared principles and practices by using more methods known by both fields and by speaking the same languages [Noor et al. 08]. For HCI, existing evolution and improvement in artificial intelligence and machine learning fields provide more meaningful and immediate adaptations, in spite of the challenges of new practical employments and applications, still resulting in the enhancement of the predictability and more user involvement.

Several matches were identified between SE and HCI fields and have promoted border crossing specially for development phases [Owen et al. 06, Seffah et al. 05, Silva et al. 12]. Several researches were conducted to advance such cross-fertilization [Owen et al. 06, Seffah et al. 05, Silva et al. 12] and
commonly they behave toward advocating the mutual benefits of exchange at development phase. Whereas the UIs are intended to evolve and to be upgraded continuously during use as well as it evolved at development time.

We argue that the benefit of an agile evolvement remains valid at runtime to improve UI proactivity. To better understand commonalities and bridge the gap between both fields, the Table 8 summarizes works on agility for adaptation development at design time [Owen et al. 06, Seffah et al. 05, Silva et al. 12] and contributes agility principles for runtime context-awareness. We outline similar practices identified in previous works [Jameson 05, Noor et al. 08, Opperman et al. 97, Silva et al. 12] in the Table 8, then we put forward similarity between fields for UI adaptation at runtime. We consider main agile practices that were valuable for the development phase and we extend their support for runtime context-aware adaptation.

### 2.4 MICAA Quality

The study of UI quality is not in the scope of this thesis, however it is still essential to manage UI context-awareness with respect to quality criteria. A fair amount of research has been conducted to identify and help developing advanced adaptations in order to streamline interaction with regards to quality criteria. In view of that, we aim to improve MICAA approach to fit interaction requirements. We focus on interaction as a key factor for improving the user satisfaction and the interface usability during use. Our goal is to improve reliability and consequently the usability of MICAA adaptation. Therefore we studied existing works and identified quality criteria for UI development.

#### 2.4.1 Quality and Adaptation approach

Several key factors account for UI quality and their usability, context-awareness and adaptations are supposed to have the major positive effect. A fair amount of research has been conducted to identify and help developing advanced adaptations in order to streamline interaction and improve its quality. Quite few studies in the literature paid attention to different adaptations approaches shortcomings and called to consider an intermediate adaptivity level [Gajos et al. 08, Brusilovsky et al. 04]. Proposed solutions were mainly devoted to overcome the transparency and controllability deficiencies while keeping advantages of automatic adaptivity [Brusilovsky et al. 04]. Three main criteria were considered in different granularities levels and ways. Context-aware-adaptation limits involve (1) allowing the runtime personalization by giving the user control over the UI design, (2) making the system predictable so that it gives consistent reaction given the same user’s feedback within the same context during the same interaction session, and (3) making the system transparent so that the user can understand approximately internal system inferences during use.
One way to achieve required implementation consisted on allowing the user a partial control on adaptation. This solution defines a MICAA approach combining adaptable and adaptive behaviors. Researchers in [Chang et al. 07, Evers et al. 12] argue for mixing both behaviors to balance out the controllability and the predictability levels. Bunt presents a literature review discussing the relevance of the Mixed-initiative UI topic and stresses the necessity of the current adaptive systems to provide end-users with an adaptive mechanism assisting the personalization process [Bertini et al. 05]. Along the same lines, MICAA similar solution was proposed by [Kaber et al. 04], it consists on an adaptable system allowing users to personalize the UI, while assisting them by recommending customization. The system reinforces the comprehensibility of adaptations. Evers [Evers et al. 12] addressed controllability in adaptive UI by integrating the users in the system’s self-adaptation loop. Both implicit and explicit controls were supported, implicit control defines the user’s influence and the explicit control allows the user to change the adaptive behavior of the application [Evers et al. 12].

Transparency was addressed from different perspectives, [Dessart et al. 10] targets the transparency via animated transitions displaying the adaptation process explicitly to the end user. The transparency was addressed in a different way by self-Explanatory UI, which have the capacity to provide the end-user with information about its purpose, structure and design. Predictability was stressed as a crucial evaluation criterion for adaptive UI [Gajos et al. 08, Gram et al. 96], existing works do not address transparency in the same way or with focused solutions.

Further we believe that the interaction quality and the UI usability concerns exceed these criteria. On the other hand, despite the authoritative nature of international standards for usability, many of them are not broadly considered. Commonly, standards are established to provide valuable tools for promoting HCI best practices. Their completeness, their relevance to practices as well as their cost/benefits are widely discussed [Bevan 09, Gram et al. 96]. To overcome such a weakness we believe that establishing standards to support interaction quality and system usability is of great importance. We argue for complete covering and support of standard principles within a well-established structure. To that end we refer to the iso9241-110 [ISO 06] to define a methodological guidance structure for supporting interaction with intelligent user interfaces.

### 2.4.2 MICAA Quality requirements

Almost three main criteria were of paramount importance to the assessment of the literature and expected to contribute the improvement of UI usability are: Controllability, Predictability and Transparency [Brusilovsky et al. 04, Bunt et al. 10, Garcia et al. 10, Gajos et al. 06, Wikens et al. 00].
By the analysis of the related works and tools, as presented and described in the previous section, the main adaptation shortcoming is still the lack of systems that reinforce and support all criteria from early development stages (in a powerful, robust, and complete manner). In this section we present main identified functional requirement:

R.1 **The support of controllability** represents the capacity of system to support user-initiated customization of the interface [Peissner et al. 13]. Many works (e.g. [Shneiderman 97][Kaber et al. 07][Kniewel et al. 14]) argue for providing user full control over automatic adaptations as a major requirement for improving adaptive systems. An intelligent interface is intended to provide users support for adaptation tasks. Users must be able to adjust their interfaces according to their preferences and needs and then being able to accept or decline system adaptation decisions. Adaptation could be controlled before (in pre-controllability, users may accept, deny or postpone it) or after (in post-controllability, users may assess the adaptation, accept or undo it) [Kniewel 14].

R.2 **The support of predictability** focuses on the extent to which past and present interface allows user to determine the outcome of future interactions, it is about actions and effects [Peissner et al. 13, Gajos et al. 06, Gajos et al. 08]. [Gajos et al. 06] considers that an adaptive system is predictable if it follows a strategy users can easily model in their heads, and evaluated predictability effects on user satisfaction. We assume that the accuracy and the predictability of the UI increased user satisfaction. [Tsandilas et al. 04] draw attention to the negative effect of inadequate adaptation accuracy on user performance in adaptive menus. Intelligent UI should maintain a high satisfaction level for their users. Accordingly providing accurate adaptation for the user context enhances such satisfaction and the trust. Accordingly it increases the subjective predictability of the system’s behavior.

R.3 **The support of transparency** concerns the ‘honesty’ [Peissner et al. 13] of system. It presents the capacity of user to understand adaptation and interpret perceived information. [Hook 09] argues for transparency as one on main usability principle for intelligent user interfaces. Only few approaches aimed to increase the transparency of automatic adaptations have been published [Frey et al. 10, Dessart et al. 11]. Dessart suggest animated transitions for viewing the adaptation process to the user. They develop a catalogue of “adaptation operations” with suitable animated transitions to support continuity in the perception of the UI changes at runtime. Other approaches aim at a deeper user understanding of the system’s adaptations by providing detailed justifications [Hook 09, Peissner et al. 13]. However, it seems questionable if and how these approaches can match perceptual, cognitive and motor impairments in users.

All above-mentioned criteria agreed on the fact that successful adaptations must prevent confusing situations and should avoid the trouble of losing control over the user interface for end-users who must be at the heart of
adaptation. Their involvement could be achieved by providing non-technical designers and typical users with user-friendly techniques for managing adaptations. Provided techniques should consider the user aptitudes at different levels: perceptual, cognitive and motor. Existing works promote above selected criteria in different ways and using different policy, however they still suffers from shortcomings and most of existing adaptation methods covers partially such concepts [Hook 00]. For instance, knowledge-based systems appear a significant lacks of control restricting the user-centeredness and deprive the user comprehending. On the other hand advice-giving systems grant user the control of selection however the decision-making process is obstructive and lacks in predictability and transparency.

2.4.3 MICAA Quality: Challenges

The interaction quality depends on the UI’s context of use and situation complexity but above all on user preferences and satisfaction. The main interaction issues could be summed within three points: incomplete user model, lack of user involvement and complex adaptation model [Peissner et al. 13]. Such limitations lead to a list of challenges that consist on:

Improving users support: Establishing an effective personalization requires recognition of the user’s preferences. However, due to the changing interest of users, it is difficult to obtain accurate and sufficient user representation from user profile and abstract user models. A new trend of user centered adaptation focuses on accruing information on users based on their interaction and feedbacks. Back to the year 1983, [Herczeg 09] defines feedbacks as “information about the gap between the actual level and the reference level of a system parameter which is used to alter the gap in some way”. This seems promising to improve their involvement in system decisions and consequently their satisfaction. The consideration of UF and learning users preferences during interaction for adaptations is intended to increase the user satisfaction during time and reduce the system complexity [Peissner et al. 13].

Improving decision-making process: The system involves several complex models that require more inferences to support advanced functionality such as acquiring, considering up-to-date contextual facts and adapting the UI at runtime. Although there were successful adaptive systems 10 years ago, they did not often make use of particular users’ preferences and context’s circumstances at runtime. A context-aware adaptation should have a crosscutting impact on the software design and appearance depending on the interaction’s context with an insignificant cost [Parasuraman et al. 00]. Handling context-aware adaptation over time considering acquired data during use is still a major UI requirement. However, adaptation decision-making is a double-edged sword. On one hand, it enhances UIs pervasiveness and proactivity. On the other hand it could increase the user’s workload and lead to their frustration. Accordingly, systems that adapt and change their behavior to
better-fit users’ requirements could disturb user, further perceptual transparency of adaptations still is not enough to maintain understandability. A smooth context-aware interaction involves three main intentions: (1) giving the user control over the system, (2) making system decisions predictable so that it always agrees with users’ expectations, and (3) ensuring system transparency so that the user can understand internal inferences. Accordingly, three main criteria should be regarded within a MICAA adaptation: controllability, transparency and predictability.

2.5 Shortcomings and Requirements

The literature review and the analysis of different perspectives of intelligent adaptation supplies allowed the identification of a set of limitations in the existing works. Most related works are constrained and focused on deeply investigating one perspective, an intelligent adaptation approach, a technological enhancement or a theoretical structure for adaptation. In this sense, the main shortcoming is the fact that the works are partial, inflexible and not integrated, thus they do not consider an all-encompassing view of intelligent runtime adaptation. The analysis of literature allowed identifying a set of shortcomings. Some of them were deeply studied in the context of this work they consist in limitations in current approaches, as for example:

S1: Lack of unification and extensive characterization of user feedbacks (interventions) during interaction. Even when systems are adapted with regard to user by considering user model and/or user profile, the user-centeredness still not achieved. User requirements could be accurately determined from practical experience rather than from abstract analysis. Analyzing feedbacks during interaction could be a good practice to improve the user-centeredness and the accuracy of context-awareness.

S2: Limited coverage of user intervention and feedbacks in mixed initiative context-awareness. User intervention through implicit and explicit feedbacks provides different degrees of accuracy and expressivity besides varying degrees of investment and commitment that need to be deeply investigated to deliver the expected benefits [Eisenstein et al., 00].

S3: Limited coverage of users preferences for runtime context-awareness. Existing works are mainly based on user models that can cover several aspect of user however they are most unable to support changing user preferences at runtime.

S4: Modest usability for mixed initiative context-aware adaptation. Existing studies of MCAA based system usability show a limited user involvement and investment in for customization [Rosson 84].

S4: Simple skills to address MICAA adaptation (fixed adaptation rules without any cognitive competency or learning method).
S5: Inextensibility of existing support addressing context aware adaptation. Most of existing support tool shows obsolescence and a lack of extensibility [Korpi 12], accordingly there is limitation in the support of changing context information.

To cope with the major deficiencies and shortcoming of existing UI context-aware system, a list of explicit requirements was identified. Based on identified limitations, we identify key improvements that show up needed technical contributions.

R1: Unified extensive characterization of user feedbacks during interaction. (C1) Feedbacks impact on adaptation should not be limited to the UI changes during interaction. It is the only mean for tracking the changing needs and interests of the end-users. It should be deeply analyzed and interpreted to assure the full understanding and an accurate identification of user preferences. An extensible feedback ontology focusing upon the multidimensional nature of feedback and addressing the runtime personalization process needs to be defined. This will allow a more accurate assessment of users’ preferences.

R2: Multidimensional unified system description supporting adaptation context-awareness. (C2) The completeness and coherence of the conceptualization and analysis tool of an approach are the only effective safeguard for well-established and successful system. An all-inclusive framework supporting the approach from different perspectives must be defined.

R3: Complete MICAA coverage: Three levels of characterization structural, procedural and conceptual. (C2, C3, C4) The method should ensure an integral coverage of involved aspects that support context-aware adaptation.

R4: Advanced logic integrating a cognitive layer for handling context-aware adaptation. (C7, C8, C9). Simple algorithms are no more efficient to support context-awareness requirements, more advanced one are required in order to improve adaptation decisions based on complex reasoning and inferences.

R5: Enhanced support for user consideration in mixed initiative context-aware adaptation. (C6). User preferences are the most relevant and/or constraint directive to define adaptation that improves UI usability. As every user interaction can contribute to an unreserved interest-indicator, involving user is a must in order to ensure a better understanding of the context circumstances and user preferences consequently enhance the UI quality and the system usability.

R6: Application-domain independent guidance for MICAA (C10, C11, C12). Systems in different domain considers adaptation as a solution for their
improvement, accordingly MICAA associated theoretical method must be able to support different interaction scenario.

**R7. High context-awareness agility for supporting runtime MICAA.** (C4). Runtime requirements as well as user needs must have a highest priorities, adaptation should be able to accommodate progressively up to date requirement and to be upgradable by learning new knowledge about user preferences.
Chapter 3  MICAA framework

The previous chapter identified MICAA important facets and discussed shortcomings in order to elicit requirements. This chapter describes conceptual choices that have been decided in this thesis. The methodology of this thesis was structured in order to meet identified requirements. Our intent is to increase UIs effectiveness by improving interaction through MICAA approach. MICAA contributes to the UIs interaction improvement by supporting the users in performing their tasks during a context-aware interaction.

UI adaptation is referred as practical depending on the level it is customized (managed, updated, adjusted) and carried out dynamically at runtime, based on contextual facts and users preferences. Context-aware UI is expected to provide a greater interaction in achieving tasks, through their capability to accommodate up-to-date requirements and novel situation through proceeded adaptation decisions. Against this background, a particular importance should be accorded to the end-user involvement when determining and agreeing adaptation. The usefulness of human interventions is that it allows the guiding, verifying and improving the accuracy of adaptation. In the other hand, system can be able to acquire new knowledge and to learn new adaptation by monitoring the users interaction. In this regard, this chapter is intended to contribute MICAA approach by providing a methodological guidance for analyzing and developing MICAA adaptation that enhances context-awareness and user involvement.

This chapter illustrates conceptual contributions defined in this thesis (Fig. 10). The conceptualization of MICAA encapsulates five main contributions. As a first step, we emphasize the importance of user involvement during interaction to support adaptation. An extensive study of user interventions and their implication in term of user satisfaction and preferences was established. The weight given to the available evidence (interventions) is mainly influenced by factors such as the quality of the data, its consistency, nature and relevance of the information for the given goal.

The analysis of existing user’s interaction flow resulted in User Feedback Ontology (UFont), ontology for user feedbacks providing a unified
understanding of user intervention via an explicit specification/analysis of a feedback stimulus. [Uschold et al. 96] [Gruber 93]. UFont is aimed to provide a formal model to clarify different interpretations and suspicions as regards end-user satisfaction.

Then, we present a Meta-model conceptualizing statically involved feature for MICAA. An associated structure supporting and characterizing adaptation process. The structure is aimed to organize involved features, conceptualized in the meta-model, within a all-inclusive arrangement factoring out and enriching the common adaptation elements. Then we establish a procedural lifecycle for MICAA outlining the cross-fertilization between UI context-awareness and agile process. The aim of MICAA-MF is to set up agility as best practice for the enhancement of the UI context-awareness beside the improvement of interaction. Finally a support structure MICAA-ISS is established. It considers interaction quality criteria defined in the ISO2941 to supervise and guide the development of a MICAA UI. It presents a methodological support for guiding and evaluating interaction quality with regard to well-defined dialog principles in ISO2941.

### 3.1 UFont: User Feedback Ontology

As discussed in the previous section, involving users through their feedback for adaptation decision is a key factor for the enhancement of interface quality in use. However the usefulness of the feedback information depends upon their aspects: (1) consistency which refers to the extent to which the feedback communicate a positive or negative judgment, and (2) credibility and accuracy concerned with how accurately the recipient (the system) perceives the feedback from source (the end-user) [Ilgen et al., 79].
In this section, we investigate user feedbacks as a means to acquire new knowledge and comprehend user preferences during interaction. Both implicit and explicit feedback exhibits different points of users' preferences with pros and cons. Users' needs and interest change continuously, enabling more users participation through feedback in order to enhance visualizations and allow (1) to offer more explanatory facilities, (2) to provide better setting of user expectations, (3) to specify deeper understanding of the context of use, and (4) to expand other improvements that considerably upgrade usability.

Both implicit and explicit feedbacks provide a new input paradigm for adaptation. Their compilation into a user preference model presents a number of challenges and can has potential to enhances the performance and accuracy of adaptation [Rosman et al. 14]. Numerous works investigate understanding and identifying subjective user feedback for effectiveness and satisfaction [Damodaran 96, Ilgen et al. 79, Ramaprasad 83]. Different techniques were identified with a common purpose of supporting the improvement of the interaction quality and its appropriateness for the end-user.

User feedback could be interpreted in different ways to suggest adaptation decisions, instructing upcoming adaptation and providing correct guidance. Given the richness of user interventions with data, there is a need for active exploration of users reactions. This will lead to a better understanding of users active opinions and behavior.

In order to provide a common understanding of the structure, credibility and information value of feedback among stakeholders (users, developers, designers) we outline a user feedback ontology enabling a formal feedback’s analysis. We believe that such ontology could offer less ambiguous specification of feedback improving their consistency and usefulness. Ontologies provide classes of objects, relationships and domain constraints on their properties. It is intended to be a semantic specification. By drawing feedback concepts within an extensive ontology, a unified interpretation of user interventions can be shared. We define three main classes to establish our ontology, the role, the significance and the modality (Fig. 11) [Mezhoudi et al. 15E].

The role outlines the feedback as an incentive. Feedback acts as an incentive/disincentive when it conveys a reward/drawback that affects system behavior and performance. Recognizing that feedback allows a user's subjective assessment of the gap between current and intended system performance, its incentive character could be achieved through four dimensions; of evaluating, valuing, organizing and correcting. Each dimension provides the user with the ability to relate to his/her personal preferences within an informative or effective manner.

- Evaluating. In this case, feedback represents a point along a continuum of goodness/badness scale. Systems are provided a transparent evaluation of their performances with respect to the user goal. Evaluative feedback is
meaningful in the purpose of usability assessment, and establishes a basis to upgrade it. It takes a variety of evaluation forms; choice when the user explicitly promotes a solution, and judgment in the case of ranking explicitly an interface feature. For instance “emoticons based feedback” aiming at expressing the satisfaction degrees among end-user, via picking an emoticon judging his/her user experience. Arhippainen [2004] gives an instance application of an answering to feedback problem with emoticons.

• Valuing. It refers to the extent to which the feedback’s valuing of information indicates goal accomplishment and performance concerns. Feedback could serve as a (reward and/or punishment) allowing to promote/demote system performances in responsiveness (more technically, as a secondary reinforcer), if, over time, the combination of feedback from certain positive and/or negative outcomes leads to carry on reinforcing properties. Knowing that the usefulness of valuing feedback depends mainly on the system ability to engage in an efficient dialog with the user. Further, the systems should be endowed with the ability to infer (analyze) such feedback and continue to learn about a user’s goals and needs.

• Organizing. Feedback serves a purpose in organization; it may be stabilization, control, growth, or change. To attend every purpose, a mindful assessment has to be made for the gap between the actual status and the expected value of a parameter, invoking or not invoking, making choice, and ordering services. The value of feedback providing new knowledge can be improved by providing efficient means by which users can directly invoke or promote the features. Likewise, they may allow end users to show their preference by accepting or cancelling the system recommendation.

Significance. This perspective denotes the focus of feedback independently of the effects. It concern the role of the feedback in putting forward a set of indicators required from users in order to determine the appropriate correlations between a particular behavior and a satisfaction level.

• Consistency: Positive / Negative. This dimension relates to the reliability of feedback. Mainly the extent to which feedback may be positive or negative. Positive feedback means a very good and satisfactory response of the user and the system. Negative feedback means a neutral response to the feature or a clear dissatisfaction. Correspondently, from the system side positive feedback could be perceived and recalled more accurately than negative feedback when they are not explicitly conveyed.

• Liability: Profits / Losses. Feedback can improve the profits of an application or the opposite. If a user reacts positively, it changes its marketing strategy and achieves progress. A negative feedback (silence, forged feedback) may be responsible for the weak and sided business. This
dimension takes into account the engagement of users in the communication and the credibility of their feedback for systems.

Fig. 11 A users’ Feedback Ontology [Mezhoudi et al. 15E]

**Modality.** Characterize the form of how a user inputs data. The variety and complexity of existing methods for eliciting feedback result in a confusing communication flow. In order to handle such complexity we proceed to analyze feedback stimulus regarding four dimensions.

Fig. 12 Interaction of users goal and feedback specificity [Ilgen et al. 79].

- **Consciousness:** Conscious/Unconscious. It represents if the feedback performed has a specific goal. When goals are general and feedback is “unconscious” the system may have difficulty in relating users preferences. According to [Ilgen et al. 79] the more specific the feedback, the more information is provided for being able to set specific goals (Fig. 12).
Chapter 3. MICAA framework

- Form: Implicit/explicit. It is interested in how accurately the feedback could be expressed and perceived. Diverse degrees of expressivity are provided depending on feedback format. Generally explicit feedback is perceived and recalled more accurately than implicit feedback. In standard explicit feedback, the user will provide ratings for items. Thus explicit feedback captures both positive and negative user preferences. On the other hand, implicit feedback can be positive. For example, if a user did not listen to a track that does not mean he does not like the track. Implicit feedback takes advantage of user behavior to recognize user interests and preferences. [Oard et al. 01] proposed a classification of implicit feedbacks technique regarding two dimensions, Behavior Category and Minimum Scope. The Behavior Category (Examine, Retain, Reference and Annotate), refers to the underlying purpose of the observed behavior [Oard et al. 01]. Minimum Scope (Segment, Object and Class), refers to the smallest possible scope of the item being acted upon [Oard et al. 01]. This classification was enhanced by [Kelly et al. 03] (Fig. 13). [Oard et al. 01] classification is limited to simple interaction, however additional behaviors could be useful (e.g. eye movement, gesture etc.). [Maglio et al. 00] implemented a Simple User Interest Tracker that tracks computer users through multiple channels to determine their interests and to satisfy their information needs.

<table>
<thead>
<tr>
<th>Minimum Scope</th>
<th>Segment</th>
<th>Object</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Examine</strong></td>
<td>View</td>
<td>Select</td>
<td>Browse</td>
</tr>
<tr>
<td></td>
<td>Listen</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Scroll</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Find</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Query</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Retain</strong></td>
<td>Print</td>
<td>Bookmark</td>
<td>Subscribe</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Save</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Delete</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Purchase</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Email</td>
<td></td>
</tr>
<tr>
<td><strong>Reference</strong></td>
<td>Copy-and-paste</td>
<td>Forward</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Quote</td>
<td>Reply</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Link</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cite</td>
<td></td>
</tr>
<tr>
<td><strong>Annotate</strong></td>
<td>Mark up</td>
<td>Rate</td>
<td>Organize</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Publish</td>
<td></td>
</tr>
<tr>
<td><strong>Create</strong></td>
<td>Type</td>
<td>Author</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Edit</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 13 Extended classification of behaviors that can be used for implicit feedback from [Oard et al. 01]. Sample of possible behaviors that user might exhibit.

[Claypool et al. 01] provide a classification of diverse interest indicator categories, and address the fundamental question of which observable
behaviors can be used as implicit measures of interest. In [Claypool et al. 01] the author examined several behaviors: mouse clicks, scrolling, and time on page. Mouse clicks and scrolling were measured both as a frequency number (i.e. number of mouse clicks) and as total time spent. The authors found that time consumed on a page, the amount of scrolling on a page and the combination of time and scrolling had a strong positive correlation with the explicit ratings. However, the quantity of mouse clicks and the scrolling measures were not demonstrated to be practical in predicting the explicit ratings.

- Simple/Complex. Feedback can be simply through a nod of the head, conveying a brief yes or no, or it can be complex as a lengthy written response. Feedback involves circling back of information to a control method to adjust behavior. For instance users could express their satisfaction levels via simple widget, or by adjusting the system sitting, and in same case the feedback could be a reporter.

- Support: Unichannel/Multichannel. This dimension provides an opportunity to define the feedback communication channel. Fisher [2001] identifies two main communication channels for UI interaction. The narrow explicit communication channel and the implicit communication channel. The implicit channel requires a considerable amount of knowledge about problem domains, communication processes, and the agents involved [Fisher 01] (Fig. 14). A feedback could be provided through a different channel. In the typical case, diverse elements included under the single rubric of feedback could share the role of conveying information about past system behavior or performance assessment which increases the feedback accuracy. However, the complexity of a multichannel feedback could be confusing rather than meaningful.

![Knowledge based Human computer Interaction](Fig. 14 Knowledge based Human computer Interaction [Fisher 2001].)

- Timing: Synchronous/Asynchronous. The timing dimension refers to the interval between the user feedback and the system reaction. When the feedback and system reaction are not consecutive may affect the learning
process. According to [Ilgen et al. 79] UI response tendencies weaken over time, accordingly delayed negative feedback is affected by less proactive interference. On the other hand, positive feedback is not affected by the interference when given immediately after the response or later, so the delay is less of an issue [Ilgen et al. 79].

Assuming that, feedback indicates the UI lack of reliability, robustness and comprehensiveness, systems proceed to analyze collected feedbacks at different stages. Gathered user feedbacks may be valuable at different stages in order to create prioritized tasks improving their product and provide a continuous assessment of product acceptance. Hence developer can decide on the way of consideration of feedbacks (immediate tactical adjustments, revenue interlocks, the long term changes).

In the case when the system allows an immediate adaptation the considered user feedback must be trustful. Accordingly it should be immediate, simple, explicit, conscious and effective, for instance, Edit, Update, Accept/Decline. However in case of long term changes user feedbacks could be collected by more complex channel such as “comment” that can be analyzed and interpreted later.

3.2 MICAA Conceptual Framework

In this section we define established design decisions for supporting the development and execution of MICAA approaches. Defined concepts contribute to the establishment of proper conceptual pattern of support tools. It is based on gathered knowledge and provoked requirement during the systematic review of literature. The proposed conceptual framework consists on three features: meta-models, structure and a methodology.

For context-aware UIs, the cost of incremental change and deployment is often higher than expected because of concerns about encoding and end-user training. Accordingly, improving the context-awareness of interface can contribute in reducing this risk by supporting the progressive enhancement and gives a good start in the right direction. Accordingly, iterative upgrades (adaptations), in view of the user preferences, will not be painful for users, and will have lower development, deployment and training costs. Thus current UIs development envisages more advanced algorithm as well as an externalization of design visions from early stages; they break with common HCI practices only considering designer’s choices to capitalize on the user as base for personalized solutions. Besides being user-centered, UI tries to converge into alternatives results, which are then continuously evaluated and refined according to user preferences.
We assume that the support of context-aware adaptation requires an agile lifecycle. In view of that, MICAA approach aims to improve context-awareness at runtime by supporting a progressive enhancement of UI and a significant user-centeredness. Suitably, we established a tripod framework tailoring agile practices to support context-awareness. Our discussion consists mainly on stressing the user involvement during interaction and enhancing the use of advanced ‘intelligent’ techniques for handling adaptation. The proposed conceptual framework capitalizes on emphasizing an agile adaptation during execution by:

- Establishing the different units that relay to MICAA adaptation: all relevant concepts and practices that characterize runtime adaptation and support diverse descriptions, implications and considerations of the approach.
- Providing an explanatory arrangement for understanding MICAA adaptation,
- Providing an agile progressive enhancement for UI by ensuring the integration of different MICAA practices and techniques.
- Addressing usability goals, as well as user experience and user performance goals by recognizing the positive contribution of user involvement.
- Supporting the new urge of researches on intelligent UI to systematically outline runtime context aware adaptation based on the advantageous idea of decomposing adaptation for evaluation purposes revealed by Totterdell back to 90 [Totterdell et al. 90].
- Providing a common ground for discussing analyzing, comparing MICAA approach.

3.2.1 MICAA Meta-model (MICAA-MM)

The established conceptual meta-model brings together concepts supporting MICAA [Mezhoudi et al. 15 D]. The reviewed literature allowed an analysis of involved concepts and their characterization. Along with above detailed requirement (Chapter 2), we propose a conceptual meta-model aimed to cover main features advocating MICAA approach. The meta-model is aimed at supporting an explicit, comprehensible and wide-ranging configuration of adaptation concerns and allowing advances and improvements. It is intended to cover the whole involved concepts and determines their relationship and dependency. Three main packages were identified to distinguish involved classes belonging to different adaptation dimensions (fig. 15).
Chapter 3. MICAA framework

Fig. 15 Adaptation main concepts

The Adaptation package is the heart and the engine of contextualization, linking all key elements involved for UI adaptation. The UIModel package defines the user interface independently of both adaptation and the context of use. The context of use package corresponds to the adaptation triggers and all the contextual factors. In what follows a detailed description of involved elements is presented.

A. The Adaptation Package

The model of the adaptation package (fig. 16) establishes the adaptation as a model separate from context and interface definitions. This dimension includes all classes related to the adaptation itself. It is intended to give an abstract but formal conceptualization for the adaptation process in terms of UI states and transitions. A UIState remains a characterization of a UI model consistent with a context assessment. The state terms values of UI attributes with consideration of the context of use. For instance at a concrete abstraction level, for a phone device context the choice interaction unit for a values number up to 30 is assessed to a Drop-down list.

UIs adapted features (for instance, Interactors, Task, AbstractUnit, Widgets, etc.) depend of the considered abstraction levels from defined UI Models and the values depend on the current context. The UIState changes during an adaptation process through a set of Transitions recapitalizing the interface changes. A transition presents a set of adaptation rules targeting a set of UI attributes and accommodating a context change.

The AdaptationRule is a part of the Transformation Model which consists on different mapping models such as reification, translation, reflexion [Calvary et al., 03]. It consists of one or more TriggerEvent initiating an adaptation and a set of actions performed to change the UIState. For example, we can imagine that an end-user could have an explicit control on the UI definition via his/her feedback. The adaptation could be also triggered automatically based on an autonomous systems’ decision-making process regarding a context change.


B. The Context Of Use package

The Context of Use has been modeled as a specialization of User, Platform and Environment. The fig. 17 gives an overview of the main entities modeled by the Context of Use. These entities are intended to identify the set of attributes and properties that will be considered for influencing the adaptation process and providing a trigger event for an adaptation. The ContextElement class follows previous works and determines the set of descriptors that can be considered to define context dimensions; in some cases of adaptive UI, features values are determined via ContextSensors.

Fig. 17 The User Model
Chapter 3. MICAA framework

As the context is a composition of information gathered regarding different dimensions, it contributes to the definition of adaptation rules conditions. The ContextElement defines the context of use as well they present the trigger for all AdaptationRules.

The UserModel class is expanded with the Feedback class and the UserProfile class. The Feedback class defines the evaluated behaviors of the user during interaction. It is aimed to enhance the user involvement during the adaptation. The UserProfile class (Fig. 17) has been modeled as a composition of Language, Knowledge, Country and PreferredRepresentationStyle.

The Language class consists of the base language used by the user. The knowledge class defines expertise level of user in using interfaces. This class can be used to organize the information on the interface. For instance, an advanced user might require less guidance to accomplish the tasks. Instead a novice user will require a comprehensible interface that will support and guide him/her to the accomplishment of tasks.

The Preferred representation style can be video, text and/or audio. Theses preferences help to the system to determine the best adaptation of the information. The meta-model outline an enumeration stating potential styles.

The Feedback class as a ContextElement allows handling adaptations priorities that must be assigned to prevent conflicts, for instance user feedbacks could be considered to evaluate an adaptation rule and promote or demote it. The Feedback is involved for different purposes, for instance control trigger and/or evaluate adaptation decisions. This class is an aggregation of the user model; it is a specialization of the TriggerEvent class as well as the ContextTriggering. The Feedbacks class is intended to assess the UIState that depends on the current context of use defined by ContextBinding class. An adaptation is triggered by a change in this context surrounding the interaction.

The Platform class determines the set of information that characterize the hardware used by the user. Fig. 18 gives an overview of the main entities modeled by the Platform. The root entity is the Platform class with is linked to the Operating System and Device. The information considered includes some characteristics of the Device and the operating system used to access the application. The Device considers integrate sensors, the screen size, the battery level, the language and the network providing the connection. For instance, a GPS that permits to acquire the geographical coordinates of the user. In the case of a battery low level, the adaptation can’t be considerate as a multimedia element.
Chapter 3. MICAA framework

The environment model (Fig. 19) provides the characteristics of the environment in which the user interacts with the device. The environment can be represented as different aspects (Time, Date, Noise Level, Movement Status, Language, Weather, Direction and Location) considered by [Sottet et al. 07, Motti et al., 13]. The climate conditions like the Weather can determine how the information can be presented on the screen. The Location of the user and the Noise Level determine the more adequate type of interaction to the user.

C. The UI Model Package

The proposal of the UIModel package can be related to any UI approach, such as Model-based approach showing a combination of UI models defined in the Cameleon reference frameworks [Calvary et al. 03] and PIM model which could be considered a model-based approach considering only the Final UI Model. The UIModel is decomposed mainly into four step organizing four abstraction levels:

- TaskModel providing a goal-oriented description of interactive systems suitable for reviewing temporal relationships between tasks, and their decomposition into elementary tasks [Calvary et al. 03].
• **AbstractUIModel** outlining an expression of a UI in terms of interaction units without making any reference to implementation [Calvary et al. 03].

• **ConcreteUIModel** presenting the UI in term of concrete interaction object that are modality-dependent, but implementation technology independent [Calvary et al. 03].

• **FinalUIModel** that represents the final implementation realized in a programing language [Calvary et al. 03].

The **UIModel** support the execution of model throughout transformation. A **TransformationModel** links involved models during generation process. Commonly transformation was merged with adaptation, however, we argue for separating them as it could improve the transformer engine performance and reduce their complexity. [UsiXml 05] and [Vanderdonckt et al. 93] identifies three main transformation: reification: form a high abstraction level to more concrete one, abstraction: from an level to more abstract one (reverse engineering) and translation which regards transformation, within the same abstraction level.

### 3.2.2 MICAA Structure (MICAA-str)

Once the static aspects and classes related to MICAA approach are established we need to outline their structural arrangement and relationship. MICAA-str addresses the low governed relation between UI adaptation features, it presents the internally organized content. The aim is to provide an extensive characterization for MICAA allowing the understanding of how adaptation concepts are considered by the system and what is made up as a result. The proposed structure put forward fundamentals of MICAA. It put forward not only the arrangement of MICAA elements but also their movement in time, their sequence and rhythm, the law of transformation of a process. So MICAA-str is actually the law or set of laws that determine a MICAA’s composition and functioning, its properties and stability.

Correspondingly, MICAA-str identifies systematically adaptation features according to Quintilian questions. Six blocks were recognized referring the context of use (To what?), the UI (Where?), the perceptive layer (When?), the adaptation engine (How?), the adaptation repository (What?) and the artificial cognition (Why?). On the other hand, it focuses on their arrangement and relations between them within an incremental and iterative process, besides highlighting the significance user involvement. We assume that those concepts contribute to the adaptation quality and maximize the chances of developing a UI with a great user experience.
As we previously presented, the main focus of MICAA approach is to advance runtime adaptation through continuous upgrading for improving context-awareness. Accordingly this structure is supposed to delineate the law of mutation of involved concepts in order to ensure an upgraded adaptation. We present our structured method that conveys a reproducible solution to design more practical, usable, and incremental runtime adaptation by making use of technological innovations for recognizing the context of use. The resulted structure extends existing work on adaptation analysis within an all-inclusive characterization structure taking a broader view and providing a unified and detailed description.

Fig. 20 Method for supporting the analysis of context-aware Adaptation

The structure is aimed to provide an extensive and extensible methodological characterization supporting different adaptation scenarios. It is intended to contribute to the development of a method supporting an explicit, more comprehensive description of adaptation concerns while also allowing their improvement. In addition, it must benefits stakeholders by providing them with consistent illustrations of design and adaptation principles. This makes possible the comparison and evaluation of the different existing and upcoming adaptations within a unified perspective as well as the identification of any gaps.
Chapter 3. MICAA framework

Table 9. Description of structure layers

<table>
<thead>
<tr>
<th>Structure Layer</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>To what?</td>
<td>Recognizes the ontological context of use, which supports reasoning about several items influencing the interaction depending on the situation of use. The UI adaptation capitalizes on the most relevant set of features and characteristics, commonly summarized by the triple &lt;User, Platform, and Environment&gt;. Moreover, information about the context items is required during the usability evaluation in order to get an objective view of interaction.</td>
</tr>
<tr>
<td>Where?</td>
<td>Recognizes the Final UI which illustrates the adaptation effects regarding the “to what” requirements. The UI adapts through the modification of interaction in several ways; in terms of interface features focusing on the way of displaying information (e.g. context-dependent menus); interaction styles and modality refers to the sensory channel used for information exchange (e.g. visual, haptic, auditory) and the interaction level that defines the amount of interaction-control accorded to users regarding their experience.</td>
</tr>
<tr>
<td>When?</td>
<td>Recognizes the involvement of systems in tracking the adaptation-triggers and supporting the decisions of adaptation. By definition, changes in the context of use trigger the system to adapt. Several works have proposed categorizations of triggers; based on these works [Knutov et al. 11] we propose a taxonomy of triggers that classifies adaptation-triggers in five categories: operator, system, environment, task and spatiotemporal. The “when?” layer considers sensors that perceive and interpret the explicit context of use (To what?), such as information related to period and localization. In addition, monitoring interaction provides more contextualization. Several studies have investigated changeable context items; for instance, user behavior in terms of implicit user feedback. Since every user interaction can contribute to an implicit interest-indicator, Unconscious interaction as well could be considered as useful data for adaptation.</td>
</tr>
<tr>
<td>How?</td>
<td>Assigns certain adaptation constituents to specific adaptation determinants, for given adaptation goals. It clearly states the executive adaptations modifying the interaction (Adaptation Engine) and the ranking mechanism (adaptation assessment manager). Those modules perform adaptation methods and techniques at the conceptual and implementation level. Performed adaptations are invoked/triggered by the “When?” layer’s events and fuelled from the “what?” layer’s acquired rules, besides they need to deal with regarding automatic situation assessment and advocate the end-user approval in order to avoid usability issues.</td>
</tr>
</tbody>
</table>
| What?           | References the adaptation strategies (rule) predefined and/or learned and acquired by systems during interactive sessions. Such rules can be easily introduced by the designer, or derived from the literature, e.g. human factors and ergonomics handbooks. They guide adaptations in existing systems, and address different levels (i.e. lexical, syntactic, semantic levels [Lim et al., 09]) of
user-computer interaction

Why? Assists the information analysis. It uses genetic learning algorithms, neural networks, fuzzy classification algorithms and other intelligent algorithms that allow diverse knowledge and outcomes to be associated and correlated, enabling the intelligent system to make reflections, process information, make interpretations and recommend solutions and decisions. This cognitive layer enhances system intelligence and provides it with the ability to decide, answer questions and explain situations via applying advanced logics and artificial intelligence algorithms.

The MICAA-str is intended to describe adaptation while supporting extension and evolution and putting forward iterative and incremental aspects. It supports the characterization of a wide range of existing adaptation approaches while maintaining harmony with intended agility. This fact is supported by users’ involvement through feedback provided during use, and also by incremental and iterative evolution through intelligent adaptation and learning of the system. The proposed characterization offers a systematic method for guiding research on advanced adaptation that supports user involvement and runtime decision-making techniques. It provides a high-level characterization of the adaptation decision-making process as well as providing guidance for different adaptation models (rule based, decision tree, Markov models, etc.).

As presented in Fig. 20 the structure factor out common adaptation elements. It depicts different modules relating to the CAA by illustrating the fundamental purposes of the involved techniques and supplies a unified illustration of adaptation strategies. Moreover it highlights the user involvement as well as the context features triggering an adaptation (To what), as well as the progressive interaction in terms of feedbacks. This provides the user with information allowing making criticism that explains his/her preferences.

By analyzing communications flows, the system extracts a set of criticisms that are subsequently analyzed in a perception layer (When). Several tracking and analytic tools are retained at this level; gathered data are conveyed to consolidate the adaptation strategies (What) progressively. Consequently the system (Where) is re-established according to the adaptation engine and adaptation Manager (How). Table 9 depicts the description structural view dimensions. Adaptations are characterized regarding considerations (to what?), implications (where? and when?) and implementations (how? what? why?)
3.2.3 MICAA Methodological framework (MICAA-MF)

In this section we put forward the agile perspective supported by the above-described structure [Mezhoudi et al. 15 A]. Fig. 21 depicts the process of adaptation putting forward the iterative-incremental aspect. The adaptation process reveals a user-centred, iterative, and incremental cycle to improve usability and consistency. Such agility involves different stakeholders of UI development, and interaction issues from developers down to end users. Agile adaptation at runtime implies that the interface adaptation is decided and performed during execution taking into consideration the layers identified for the above-described method. The process described above adopts a user-centered perspective, enhancing the intuitiveness and consistency of adaptation decisions. The Agile adaptation process consists of a divergent and a convergent phase. In the divergent phase, multiple adaptation triggers (When?) and adaptation strategies (How?) are invoked and presented for evaluation. During one iteration the adaptation inference considers only those portions of the user interface that are being evaluated. The convergent phase (Fig. 21) concerns the (What?) layer. It results from evaluations of alternate adaptations that are used to create an ascertained adaptation rule that is likely to be endorsed by user feedbacks.

![Agile methodology loops for adaptation support](image)

Gathered information is processed through the artificial cognition processes (Why?). This layer provides the cognitive structures; it is responsible for the cognitive functionality of perception, consciousness and information processing. For each iteration, the user behavior is tracked and then users are notified about execution details and adaptation progress. Users are able to add their preferences in terms of implicit and/or explicit feedbacks. This can be
advantageous because the adaptation decisions have more consideration of user choices; therefore well-defined adaptation decisions are able to create the interface closer to the end-user expectations.

The procedural arrangement provides the adaptation process with improved user-centeredness, interactivity and responsiveness to user feedback, outlining the advantages of integrating agility with context-awareness.

Thus, combined with an agile process, adaptation is enhanced by several advantages that meet present-day requirements:

- Regular and immediate adaptation to changing circumstances approved and carried out within a user-centered paradigm.
- Simplification and better understanding of the adaptation.
- Runtime testing and validation that provide immediate visual enhancement, as well as improving usability for end users.
- Active ‘user’ involvement throughout the adaptation-decision during execution.

3.2.4 Discussion

As we noted in previous section, we consider an agile method to support UI adaptation at runtime. This section aims to illustrate how the cross-fertilization of agile method and context-awareness can be of benefits for both HCI and SE communities. We believe that the proposal could be used to support the Online Methodological Prototyping (OMP) with a high-fidelity level [34] for both fields. Commonly this task consists on defining the system as small releases evaluated by users and enhanced iteratively. Each iteration
Chapter 3. MICAA framework

results new requirements that require additional cost (time and working development). This aspect remained a serious shortcoming of OMP.

Agile adaptation could contribute OMP and overcome costs shortcoming by advancing adaptation to upgrades systems. Enhancing adaptations and UI context-awareness initiate the need of deeply revising the current adaptation practices to account for (1) the alternative designs requisite for adaptation in the interface layer of system, (2) the parameters involved in driving adaptations (patterns, models etc.) and (3) the logic of adaptation at run-time defined by the operational core of the system.

Adaptations are managed by end-user during execution. Interaction session's results new knowledge that present next iterations requirements. Appropriate adaptations are evaluated and endorsed by end-users regarding their preferences and satisfaction. An agile OMP with high-fidelity level supports vertical, horizontal and diagonal prototyping [Vanderdonckt et al 07]. Fig. 23 represents the OMP dimensions. The horizontal prototype recognizes functionality that concern interface for instance by changing appearance and interaction style, colors, widgets and their arrangement (1). The vertical prototype targets more deep levels of system. For instance prototyping abstract models by allowing users to define and update explicitly their profiles in order to accommodate the appropriate adaptation regarding their preferences(2). As well vertical prototyping could concern operational core layers for instance by allowing end users to change the system complexity according to their expertise levels and evolution (3).

The diagonal prototype combines both above stated strategies. In this case the system can learn to adapt. For example, by monitoring users interactive behaviors system can adapt deeper layers such as updating abstract models (interaction models, user profiles, design patterns) and/or upgrading the operational core by learning new adaptation rules, making adaptation decision and extending existing functionality.

As well agile adaptation could support the evolution of systems based on user experiences. For instance for an interface with vocal modality, the system needs to learn about users details during interaction (recognizing sounds, timber, volume, learning commands). In this case, agile adaptation support the training phase to enhance the system iteratively with regards to the users expectations.
Chapter 3. MICAA framework

Methodological Guidance

A wide range of scenarios can be envisaged to make use of the structured method. We recommend a practical guide for designing UI adaptation aims and requirements, creating, developing, and maintaining specifications expressed as in the given structure. This structure presents in detail the stages followed by developers and designers in carrying out adaptation specification. The recommended scenario consists of a typical path toward making clearer presentation of the conceptual approach (Fig. 24).

For each stage, we list key questions to prompt discussion and reflection together with additional information and suggestions for improving
adaptation practices. Table 10 summaries questions about design decisions for each stage.

Table 10. Analysis of methodological stages

<table>
<thead>
<tr>
<th>Stage</th>
<th>Design decisions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Where ?</td>
<td>Where adaptation will take place?</td>
</tr>
<tr>
<td></td>
<td>Which granularity levels adaptation could be applied to (change the UI completely, partially)?</td>
</tr>
<tr>
<td></td>
<td>Which interface features would be affected by changes (functionality, modality, appearance, such as interface’s colors and sizes, interaction scenario, interaction units, etc.)?</td>
</tr>
<tr>
<td>To What?</td>
<td>Which elements of context of use are envisaged to trigger and tailor adaptations?</td>
</tr>
<tr>
<td></td>
<td>Which user criteria are to be taken in account? Such as: preferences (e.g., to personalize the interface representation), expertise (e.g., to decide the interface complexity level) or physical ability (e.g., to select colors in case of color blindness)?</td>
</tr>
<tr>
<td></td>
<td>Which devices are targeted and how interface could be displayed?</td>
</tr>
<tr>
<td></td>
<td>Which conditions of the environment may influence the interface definition? For example: luminosity affects color choices, noise influences the volume?</td>
</tr>
<tr>
<td>What?</td>
<td>Which techniques could serve to realize adaptation? For instance which practice could be deployed to migrate interface from graphical to vocal modality (patterns, models, rules etc.)?</td>
</tr>
<tr>
<td></td>
<td>Which implementation allows rearranging UI interaction units regarding end user preferences during use?</td>
</tr>
<tr>
<td></td>
<td>What knowledge and how this could be gathered and used for adaptations?</td>
</tr>
<tr>
<td>How?</td>
<td>How decisions are made?</td>
</tr>
<tr>
<td></td>
<td>How conflicts could be resolved?</td>
</tr>
<tr>
<td>When?</td>
<td>Which data could notify changes triggering the adaptation?</td>
</tr>
<tr>
<td></td>
<td>What are adaptation triggers (context data, users etc)?</td>
</tr>
<tr>
<td></td>
<td>To what extent gathered data could influence adaptation and change them?</td>
</tr>
<tr>
<td>Why?</td>
<td>How information is processed?</td>
</tr>
<tr>
<td></td>
<td>To which end information are processed (interaction quality)?</td>
</tr>
<tr>
<td></td>
<td>Which techniques could be used to help a decision-making process?</td>
</tr>
<tr>
<td></td>
<td>How the system could infer acquired knowledge and make up-to-date decisions (for instance decision tree, decision matrix, fuzzy logic, etc.)?</td>
</tr>
</tbody>
</table>
As a first step, stakeholders should specify where adaptations are intended to be applied. This phase should notably enable questions relating to adaptation outcomes. Questions raised are described in Table 10. Once the scope of the adaptation is established, designers have to specify the adaptation’s considerations (To what?). At this level, the designer defines the relevant facts that should be considered in adjusting the interface.

The contextual elements cast light on the design of adaptation rules and are a practical way of quickly retrieving the adaptation conditions subsequently. The (What?) step addresses a functional part of adaptation and the possibility of combining and using knowledge and know-how in order to assist planned adaptation and achieve expected results.

The next layers represent the functional dimensions (How?, When?, Why?) forming the core of the adaptation decision-making mechanism in the instantiated approach. At the (How?) levels designers should specify how adaptations are assigned and computed.

The (When?) step raises questions about adaptation triggers. In the case of designing an intelligent adaptation where the system is intended to be proactive and have the ability to decide adaptations, designers should provide a module supporting the Why layers. This methodological guidance identifies the steps taken in conceiving an adaptation approach at design time. The next section shows an instantiation of structure at execution and characterizes the evolution of the adaptation decision-making process during use.

In the proposed structure and associated methodological guidance, the user involvement was emphasized through the support of different form of contribution, which implies that most of the times the user will be responsible for triggering or deciding upon an adaptation process. The above presented feedbacks ontologies help in sitting, evaluating and understanding user role within the adaptation process.

### 3.3 MICAA Interaction Quality Support (MICAA-IQS)

We aim at improving MICAA by considering effectively usability principles. As discussed in the chapter 2, this can make improvement and enhance the user experience and maintain UI usability within the growing complexities of interfaces functionality and their changing contexts. Putting forward user as the heart of interaction and adaptation process lead us to support human behaviors for improving the reliability of interfaces and the consideration of dialogue principles ISO9241-110 to streamline the interaction
A set of consistent design decisions is defined to support MICAA UIs (Table 11).

In order to support human behavior, we consider Rasmussen’s model, which has been extensively used over the last two decades for human behavior modeling. The model rationalizes the Human (user) behavior controlling a complex dynamic system. This model sums up human performance within three levels of behavior: skill, rule, and knowledge. The purpose is not only to advance adaptation; it is to address the challenge of usable adaptation, taking into account different user needs with an integral prospect.

We assume that meeting users requirements and preferences effectively and efficiently during interaction should consider a conjunction of above detailed principals and concerns. Interaction is intended to maintain usability while keeping full visibility of user performance and behavioral model. The main purpose of this work is to make a step toward usable intelligent UIs. It is aimed to provide system designers with a tool (structure) to help the development of intelligent interfaces that invokes a good representation among users. This tool consists on a guidance that allows bridging the gap between user expectation and system decisions during interaction in order to support usability, improve reliability and enhance user satisfaction.

### Table 11. Association between requirements and respective Design Decisions taken

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Design Decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1: Controllability</td>
<td>DD.1 Support of user behavioral model: Processing layer: Improving the controllability is not only by allowing user intervention to personalize interfaces, but also conflicts are solved with control system and conflict resolution mechanisms. The aim is providing user an adequate interaction, accordingly controllability feature depend on end users skill and ability etc.</td>
</tr>
<tr>
<td>R2: Predictability</td>
<td>DD.2 Support of user affective model: Executive layer: Regarding the rule engines, to provide inference and reasoning, user preferences and expectations should be considered</td>
</tr>
<tr>
<td>R3: Transparency</td>
<td>DD.3 Support of user cognitive model: Perception layer: Varied complexity levels must be supported for users with different expertise levels and various knowledge.</td>
</tr>
</tbody>
</table>

We assume that usability improvement relies to harmoniously integrating controllability, predictability and transparency above described. Human reliability during interaction could be achieved via a two different ways:

- In the anticipatory stage, as a post analysis of the potential situation of interaction and as assessment of the interaction quality.
• In post-interaction, to comprehend and recognize involved features that influence human performance during interaction in order to increase users’ satisfaction and human errors.

Satisfaction is then an obvious consequence. In particular, there was a need for further development regarding the integration of quality and human aspects for one exposure scenario. To that end, the proposed structure reflects Rasmussen user performance model within interaction principals. Further, such principal are endorsed by relating to ISO 9241-110 dialogue principals for designing interaction. The advantage of this integration is the particular importance according to the end-user significance and/or involvement when determining and agreeing context-aware interaction with intelligent system. The human support during adaptation allows guiding, verifying and improving their accuracy rather than the improvement of system intelligibility to meet user expectation. System should learn through interacting with the user and its environment otherwise, it would only repeat its mistakes. Different techniques support system intelligence such as learning from observation, and from knowledge’s specified by designers.

Fig. 25 Context-Aware Intelligent Interaction Architecture: based on user's cognitive model and usability features.

We refer to cognitive aspects of user’s performance and we consider three levels to model user behavior: skills, rule and knowledge (SKR) defined by Rasmussen. Considered layers (SKR) enhance the user-centeredness and
human reliability of the method. We define the adaptation process with a full coverage of the triplet. Three level are identified:

The first layer holds *Skill feature* (control theory), it regards the concrete interaction flows allowing user to act and access the information. From the system side it denotes the capability of system to support users’ interventions. As well, it presents the ability to take such intervention into consideration for the improvement of adaptation performance via the advanced algorithms allowing knowledge learning. Several solutions targeting controllability were conveyed and reviewed in the related works; most of them focus on the user feedback. This level has to do with controllability, errors tolerance and suitability for situation (context-awareness) principals.

The *Rules layer* regards the Executive layer of users. This layer concerns predictability and human situation assessment and decision making process. On the system side the implementation of decision-making algorithm and optimization strategies is responsible for the management of contextualization. The aim is to ensure greater convergence between human reasoning and the decision-making algorithms in order to prevent distortions of users and fraud. Accordingly predictability could not be seen only as a consequence of controllability, but require to be investigated in term of deduction, reasoning and problem solving algorithms. At this layer mainly two-dialog principals are to be considered: the suitability for the task and the conformity with user expectations.

The *knowledge layer* is aimed to ensure a common understanding of adaptations and interfaces changes. From the user side, the Perception layer is responsible for this feature. It concerns information processing blocks and refers to the acquisition of incoming information for instance comprehending, relating, grouping etc. The UI support this feature within different perspective. The main intent is to insure that users correctly interpret perceived information. We argue that a transparent adaptation improves accessibility among systems.

Within this prospect, we established the methodological structure view supporting a user-centered adaptation with regard to usability criteria. An approach supporting the aimed synergetic adaptation will be convoyed. First it solves the controllability issues and reduces the feeling of loosing control resulted from system-initiated adaptations by using different users feedbacks. Then it support a ML based adaptation management algorithm putting forward predictable solutions and avoiding confusing inconsistency. This interaction is held by a simple and comprehensible graphical representation supporting transparency.
Chapter 4 **Identification of the Abstract-User Interface**

Chapter 3 detailed our approach for analyzing, conceptualizing, and creating MICAA of UIs. In order to validate the framework, a first MICAA methodological instantiation addressing the Abstract User Interface (AUI) is depicted in this chapter. M@rina consists in a model-based approach for mixed-initiative identification of Abstract User Interface, decomposed into Abstract Interaction Units in a flexible way while ensuring context-awareness. Section 4.1 defines this sub-problem and reviews the underlying aspects; Section 4.2 presents the model-based approach for AUI identification and its support by M@rina, Section 4.3 reports on the results of an experimental study testing the impact of M@rina.

4.1 **Definitions and analysis of the sub-problem**

Among existing approaches for frameworks’ evaluation [Bardram 05, Scholtz et al. 04], we selected the prototyping approach to validate the applicability of theoretical framework for different purposes by stakeholders’ (e.g., analyzing, comparing, designing). Prototyping supports the understanding of MICAA’s concepts and its potential to support agile, mixed-initiative adaptation at run-time. This section presents a first prototype for this purpose.

4.1.1 **The problem of AUI identification**

Once a task model has been created for a particular interactive task, the next step in forward engineering (reification) according to the Cameleon
Chapter 4. Identification of the Abstract User Interface

Reference Framework [Calvary et al. 03] consists in identifying an appropriate Abstract User Interface (AUI) from this task model. After the task model, which is supposed to be independent of any technological space and any context of use, any AUI should be designed for a particular context of use while remaining independent of any technological space. At this stage, no choice is made yet about the target development environment and the interaction modality.

AUI identification is aimed at enabling designers to specify a first UI that does not commit yet to any technological decision. According to [Breiner 11], the goal is to have an abstract UI that supports high-quality access to the broadest possible range of targets for the broadest conceivable range of delivery contexts, via the broadest possible set of access mechanisms [W3C]. Several AUI candidates could be generated from the same task model that exhibit various levels of qualities depending on their perspective [Breiner et al. 11]. Probably the most important quality criterion is that any AUI identified should be adapted to the target context of use. When the context of use is determined at design-time and would not change, this AUI identification is more conditioned and straightforward than when the UI could switch to multiple contexts of use at run-time. In this situation, one AUI is not enough to be adapted to the changing context of use.

The first sub-section reviews and analyses the literature related to the AUI identification by classifying algorithms into two classes: systematic AUI and prospected AUI engineering. A detailed depiction of the AUI is presented in the Appendix A.

4.1.2 Exploration tracks

Criteria for AUI identification include, but are not limited to: task compliance, consistency with task structure, adequacy to the user such as user workload, adequacy to any technological space, level of guidance, and AUI structure.

A first class of algorithms for AUI identification, called Systematic algorithms, consists in defining a mechanism for automatically generating either a large set of potential AUIs or the complete set of possible AUIs in order to support some systematic explorations of design decisions affecting AUI identification. For instance, [Coninx et al. 03] [Tran et al. 12] presented algorithms for systematically generating all potential AUIs from a task model, whose number is a factorial of task elements and temporal operators. Systematic algorithms suffer from the following shortcomings: AUIs are identified based on purely structural properties of the task model (sometimes
only the underlying tree, sometimes the entire graph including the temporal operators), AUIs do not change according to the context of use, there is no adaptation possible and there is no guarantee that any AUI resulting from this process will be adequate or valid.

A second class of algorithms for AUI identification, called prospective algorithms, addresses the problem of AUI identification based on functional and non-functional requirements of the UI such as: guidance, consistency, workload, and security. It is expected that taking these requirements into account during the AUI identification will positively influence the quality of identified AUIs, in particular their adequacy to the initial task model.

Depending on these requirements, a wide spectrum of algorithms could be introduced. We hereby report only on some feasible solutions by referencing significant examples from the literature considering different quality criteria.

• **Level of guidance:** The extent to which a product used by specified users meets their needs to achieve specified goals with effectiveness, productivity and satisfaction in specified context of use. (ISO/IEC 14598-1). Several systems are developed to assist end users during interaction but the level of guidance should vary according to the user’s expertise and other factors, which impacts the AUI identification. For instance, the level of guidance could change the AUI in terms of task repartition beyond the ability of adding, modifying, and deleting tasks for flexibility. MOBI-D [Puerta et al. 97] suggest possible AUIs according to human factors by optimizing the structure of Abstract Interaction Units in decision support system based on decision trees and inference rules. This algorithm has been developed in 1997 for a particular technological space, thus stemming for a significant update of the usability knowledge encapsulated in the decision tree and inference rules [Puerta et al. 97].

• **Consistency:** A consistent application and user interface always follows the same general principles, the same interaction principles, uses the same terminology, and so on. Consistency can also be extended over several applications, a whole operating system, or even across computer platforms (ISO 9241). A typical type of consistency consists in using standard UI toolkits or UI builders and ensuring that the generation rules conform to UI guidelines for this technological space. ROAM [Chu et al. 03] identifies AUIs for different technological spaces by addressing three types of consistencies: task consistency, layout consistency, and transformation consistency. Nevertheless, the system still has a real-time constraint of migration latency [Chu et al. 03].

• **Security:** it denotes the property that data or services are protected from unauthorized access as well as the interaction and user data. A usable UI
could be assessed as non-secure and a secure UI could be assessed as unusable. In order to accommodate both quality factors simultaneously, i.e. security and usability, therefore becomes a new requirement. Basin [2010] studies security-aware GUI by exploring the link between visualization and security: a model-driven engineering approach produces an abstract definition of security-design models, that is independent of any specification language which is in turn expressed in a distinctive language (SecureUML+ComponentUML) for modeling security-design models simultaneously.

- **Workload**: Reducing the required workload capitalizes on improving the conformity to the users’ expectations which demands that an application behaves as users expect it to do. This principle goes beyond mere consistency, because it is not restricted to the computer systems but also connects the application to the real world platforms (ISO 9241). Since a task model expresses all the activities that an end user may carry out to complete the task regardless of his/her profile, there is no consideration of task workload in this model. Consequently, every identified AUI should exhibit an appropriate workload allowing the estimation of realistic physical, cognitive, and mental workload [Beiferno 85]. Associating a workload for task should improve the UI definition; tasks could be composed, decomposed, or assembled so that the UI demand never exceeds the users’ abilities and do not overload them. Several methods exist for measuring or predicting the task workload, such as the NASA-TLX [Hart et al. 88], the cognitive walkthrough based on action analysis [Rizzo et al. 97], the Hart & Staveland metric [Hart et al. 88], and the German load index. These methods interpret the task-related workload as a metric that is either predetermined depending on the task type or reported by end users in an experiment. In both cases, this interpretation is relative and restrictive, thus potentially it does not form the basis of an overall measure for generating AUIs. However it can offer guidance and help for the generation of valid AUI.

### 4.1.3 Motivations for AUI identification

AUI identification should decrease the complexity of information systems, developed for a particular task, and guarantee the validity of their results. A high flexibility remains of high importance to support context-awareness. Different metrics related to the UI development and evaluations were advanced, however a notable lack of tasks complexity appraisal is still perceived. Weighting task is considerably valuable for modeling and adapting the UI rather than their evaluation. Most of metrics were subjective since the assessment of a task depends of different external factors such as user capability, and context of use. Assuming that (1) weighing task in the
evaluation phase was incredibly advantageous for analyzing systems in different field (e.g. military, laparoscopy, healthcare...); and (2) that it has the same importance and usefulness to make use of it to guide the UI definition, we investigated here the workload measurement which offers an appropriate measure for task. Currently the evaluation of workload is a key point in HCI researches. It puts forward a decisive-summative tool for supervising the intention of adjusting systems and learning environments. Besides it is investigated for real-time feedback for adaptive systems. Several works define workload as the physical and/or mental requirements associated with a task or combination of tasks (see Appendix C for a comparison of these metrics). We focus on a score-based task appraisal as a valuable criterion for weighting UI tasks and to guide UI adaptation.

4.2 The proposed solution: M@rina

We present our solution capitalizing on some factors supposed to improve the adaptation the AUI. The use of the model-driven approach for developing adaptive UIs enables us to apply different types of adaptations on the various abstraction levels. However, practically implementing adaptive model-driven UIs requires tools that not only support modeling, generation, and synchronization of abstraction levels, but must also include features to support the adaptive behavior of UI at runtime. Quite few of existing tools involve adaptivity within a model-driven engineering approach by rendering interfaces on multiple types of platforms, or by considering specific user capacities and disability.

Our proposal M@rina (Mixed-initiative @ Runtime INterface Adaptation) distinguishes itself by the particularity of imposing a specific reflection on the dynamic architectural model of adaptation in order to guarantee a harmonious integration between the MICA adaptation and the UI generation at runtime. The system is intended to instantiate MICAA, considering the user feedback at AUI level (Fig. 26). This feedback could be gathered implicitly and explicitly during interaction. Gathered feedbacks participate immediately in the customization of the interface.

Furthermore, M@RINA proposal puts forward an explicit support for the controllability, which should enable the user to control how the UI is adapted at run-time by explicit adjustment interface of adaptation parameters. The explicit adaptation implies external guiding principles such as usability and ergonomic guideline, which are adjusted explicitly to move the UI towards a desired status. However implicit adaptation relies on some built-in capability of
the system to react autonomously. We assume that perceived events during interaction, such as user errors, users personalization etc, increase the system predictability to perform adaptations matching the users’ expectations, in addition to reducing the potential for unexpected adaptions, users’ frustrations, and the loss of interaction.

Fig. 26 MICAA based AUI identification: M@rina instantiation

The proposal consists in a task-based UI generator enabling extensive mixed-initiative adaptations. Adaptations consider both contextual fact and users’ feedbacks (to what?). Mainly three different feedbacks means are considered: the controllability feature and the star classification as explicit feedbacks. And, user’s actions as implicit feedbacks for deducing user’s interaction patterns (When?). The most significant new feature is the partial reification process (How?). Adaptation decisions are computed dynamically regarding runtime contextual data and executed at run-time. As a result, the set of containers (windows) composing the UI are generated and adapted at runtime considering latest contextual information in a flexible way. The reification process does not generate a complete UI; it is limited to generating the next container with regard to the context of use and the displayed container. Given that most of design time generated UIs fail to provide the required flexibility, it is important to improve the UI ability to adapt and to involve the end-user and context facts during interaction. In view of that, M@rina, through applying MICAA concepts, enables during the same interaction session; (1) the detection of context variations,(2) the decision-making of new adaptation and (3) the generation of a new adapted UI.

This MICAA based adaptation led us to introduce the concept of at run-time partial reification process triggered on the user or the system request
Chapter 4. Identification of the Abstract User Interface

(Fig. 27). The UI is reified, in a partial and iterative way, taking into account novel context changes (e.g. display size, user preferences), which allow the deployment of new adaptation decisions for current and next containers. The M@rina structure aims to avoid predefined adaptation and consider current context fact while creating the next container. However, previous interaction will be considered in order to ensure a coherent interactive session. Furthermore, context-aware reification involves plenty of options for arranging interaction units, and then the selection of adapted solutions needs to be guided by a set of quality measures. Different scores are defined and assigned to each result in order to manage adaptation and to handle the discrimination of better solutions.

Fig. 27  M@RINA partial AUI Reification.

Fig. 27 graphically depicts the partial reification process. It consists on a partial UI generation by reifying next predicted action with regard to the context and user’s interaction pattern. To define user’s action patterns, we proceed with the Markov Model for the user behavior monitoring and the prediction of user behavior. We can summarize the process of user behavior prediction with the Markov chain in three steps: (i) generating and monitoring sequences of actions, (ii) learning an N order Markov chain model (or all the order from one to N), and (iii) predicting the next most probable action of the user considering the history of interaction. Accordingly the adaptation approach outlines the advantages of:

• Supporting mixed-initiative: both system and end-users are involved in the adaptation.
• Learning system: The adaptation process takes into account the evolution at runtime and capitalizes on collected data and the context of use.
• Enabling and facilitating controllability: The controllability feature ensures more flexibility for adaptation by allowing the end-user to control and adjust the involvement degrees of features, as well as choosing adaptations.

• Enhancing the granularity of adaptation: This is shown by the partial reification allowing the adaptation of UI during the same interactive session.

4.2.1 Algorithms

In this section, we make clear the functioning and usefulness of the above-specified technique for partial reification (Fig. 27). In our case, the AUI engineering starts with a triggering event and finishes with a reification of the novel adapted partial FUI. All adaptations capitalize on user feedbacks provided in an implicit or explicit way. The algorithm acts in two distinct modes for AUI adaptation (automatic and controlled); both adaptations are computed regarding quality criteria and metrics.

During a simple use, the adaptation is triggered through simple navigation between windows during the interaction. In this case, adaptation is mainly computed considering scores adjusted regarding the user behavior prediction (more details in the modeling cost function section).

The second case consists on explicitly trigged adaptation. M@rina provides a flexible adaptation of UI through direct user interventions. It provides users a controllability feature recommending appropriate adaptation for the current context. Fig. 28 shows the implementation of the controllability feature.

A. Controlling adaptation

An important goal of M@rina approach is to support controllability. Thus we put forward an adaptation recommender allowing users to manage and/or to adjust adaptation at runtime. We deployed “ML techniques” and a scores-based decision-making process to guide and control adaptation during the reification process.

Deployed techniques are intended to give models (or even end users) the ability to manage context and to fully control the adaptation process via direct manipulations. The decision support tool consists on a graphical UI to precisely define the scoring parameters and weighting them. A controllability feature (Fig. 28), is put forward as an extension of the final UI, where a scoreboard of reflected adaptation parameters is displayed allowing the adjustment of their relevance for adaptation. Parameters’ control consists of
adjusting the weight attributed to each sub-score in order to balance their importance in the final score.

Fig. 28 M@RINA GUI for supporting controllability: It consist on a list of recommended AUI ordered with regard to the scoring function (AUI profit) and in the main screen a configuration screen allowing the tuning of the list of parameters considered by scoring function.

The controllability feature present a list of tunable parameters, allowing end-users to define the weights associated with each sub-score in order to balance its importance in the final score. Considered parameters revealed in Fig. 28 are defined as follows:

- The accuracy of considering the completeness of the task model
- The accuracy of user guidance by behavior monitoring, which is used for the score of actions given the user behavior prediction (U.B.P)
- The level of consideration of Feedbacks
- This locality provides an opportunity to consider or discard the content of the previous containers in the design of the next one
- Concerning actions that appear in previous containers but which are not filled
- This locality provides an opportunity to control navigation regarding the accomplishment of tasks
- Graphical connections between widgets implementing actions belonging to the same parent task in this reification process
- The platform weight that defines the maximal number of elements displayed on the screen
On the left-bottom border, the controllability feature recommends potential AUI of current container to users (Fig. 28). An ordered preview of the best-scored interfaces is shown. By moving the mouse pointer over these buttons, the preview is displayed in the center frame replacing the scoreboard. For visibility reasons, the two first solutions were zoomed in on the left side of Fig. 28, labeled with their appropriate scores (AUI profit).

All displayed interfaces profits (score) are calculated according to weights associated for different feature displayed in the scoreboard. The scoreboard has a responsive design in which the number of proposed alternative solutions and the platform size feature change according to the window size. A simple click on the preferred preview triggers the reification of associated AUI to end the adaptation.

B. Learning and predicting user behavior: Markov chain

Personalizing the UI requires to infer user’s knowledge and intention throughout interaction in order to reach a particular solution that meet user expectation. The main benefits of gathering interaction data are making adaptation decisions and help users to better carry out their task by predicting their behavior. Markov Chains have been employed to improve the efficiency of adaptations, to manage frequent updates and upgrades and to ensure online repairs and personalization.

Various Machine Learning techniques could be deployed in different ways with a common purpose: predicting a user’s future actions [Motti et al., 13]. The proposal operates the Markov chain for predicting the action that user will take next given already performed sequences of actions. A user profile is defined in order to link information about the user to expectations concerning future behavior.

Markov models are represented by three parameters < A, S, T >

Where:

- A is the set of all possible actions that can be performed by the user;
- S is the set of all possible states for which the Markov model is built;
- T is a Transition Probability Matrix (TPM), where each entry tij corresponds to the probability of performing the action j when the process is in state i.

In more complex models, the predictions are computed by looking at more than one action performed by the user. And then the approach is generalized to the Kth-order Markov model, which computes the predictions by looking at the last K actions performed by the user. However, these higher-order models intensify the weaknesses associated with high state-space
complexity, reduced coverage, and in some cases leads to poor prediction accuracy.

To address this weakness, Longest Repeating Subsequence (LRS), which was combined with the Markov chain, is used in order to handle both spatial and temporal complexity. These complexities will be more precise if we only use Markov chain with entire sequences. The accuracy is quite the same in both cases [Mitrovic et al. 07]. The Longest Repeating Subsequence [Pitkow et al. 99] is defined as

- The subsequence is a set of consecutive actions (following each other in time) from a sequence,
- The subsequence is repeated more than T time in the sequences (generally T=1),
- We only keep the longest subsequence (the LRS that is contained by bigger LRS does not fit).

Accordingly, combining both methods results in a quite intuitive technique offering good performance. It matches the problem of User behavior prediction which aims to predict actions, based on an observed history. Moreover, in our case we are not using various types of features in order to predict some other kind of data (like the case for weather prediction [Langlay et al. 05], speech recognition [Juang et al., 91] and image analysis [Aas et al., 99]). Only actions feature type is considered. As a result, this prediction does not need complex optimization techniques, like Gaussian or Neural Network, with gradient descent at the learning step. The prediction time (using the model, query prediction based on an history) is fast since we just have to take the higher probabilities given the history.

M@RINA uses the user behavior prediction to improve the arrangement of abstract interaction units at the abstract user interface level within containers. The implementation capitalizes on a ML technique based on statistics in order to predict the next action(s) that will be accomplished by the user given the previous ones she filled. The prediction is realized via the UserActionPrediction class (Fig. 29), and can be seen as an extension of the context of use where the data are processed to extract more useful data. This class needs an instance of ActionMonitoringDB as “raw material” and also takes the Markov order as parameter. We can summarize the process of user behavior prediction with the Markov chain as follows:

- Generating and monitoring sequences of actions.
- Learning an N order Markov chain model (or all the order from one to N).
- Predict the next most probable action of the user based on his/her history of immediate action.
Chapter 4. Identification of the Abstract User Interface

C. User Monitoring and Feedback gathering

Above-detailed controllability feature is not only an explicit feedback tool, it is also an adaptation recommender. M@RINA considers both explicit and implicit feedback, aside with the controllability feature to compute and decide adaptations. Adaptation’s details can be acquired by asking the user during the interaction in an explicit way, which shows generally more expressiveness than implicit feedback, but that does not neglect its importance.

Fig. 29 a) Explicit feedback implementation: rating scale. b) Classes managing the user feedback: Partial view

Both explicit and implicit feedbacks were used to provide information about the current context of use and the appropriate adaptations. The combination of these two types of feedback provides another paradigm for adaptation. Beside the above discussed MICAA’s advantages, their combination in a user preference model presents a number of challenges and above all the performance and accuracy of adaptation decisions [Rosman et al. 14]. First, implicit feedbacks are acquired through a rating scale technique (see Fig. 29a). This feedback technique aims at allowing the user to evaluate proposed containers in order to promote/demote the score of proposed AUI.
Moreover, the feedback techniques provide a way to assess the recommended UI and determine the quality gained/lost by considered adaptation decision.

The implicit feedback is taken into account by monitoring the user experience when using the system in order to predict his/her behavior. The class UserActionsPrediction takes advantage of Markov chains to define the Longest Repeating Sequences (LRS). The class is related transitively to the contextOfUse class via the Action-Monitoring classes.

It is important to distinguish the feedback features and the adaptation’s recommender of the controllability features. Users are involved through their feedbacks in evaluating system recommendation. Displaying a recommended AUI by user implies promoting the solution implicitly. The controllability’s feature gives us the opportunity to decide when and how to define the adaptation and to choose among different recommended adaptations. On the other hand, the feedback allows to perform an “Evaluation of adaptation” regarding user preferences. It asks the user how he feels about the result of the adaptation.

**D. Modeling Cost function**

We believe that UI engineering is not only about having UI that supports high-quality access to the broadest possible range of targets for the broadest conceivable range of delivery contexts, via the broadest possible set of access mechanisms. However it is also intended to build more context-aware UI improving the interaction’s quality and increasing the usability level.

Therefore the consideration of different aspects guiding and controlling solutions are becoming a best practice and a requirement. As it was pointed out previously, the assessment of generated interfaces’ quality is determined through a cost function. Those scores provide quantitative metrics to enhance both the consistency and optimality of recommended solutions.

In what follows, we present costs and scores considered by M@rina for the UI assessment. Defined scores are computed at runtime to be an input to determine next container and consequently adaptation decisions. First score defined for gauging a UI is labeled ‘ContentScore’. This metric concerns composite abstract Interaction Units (IU) (containers) without considering the arrangement of simple interaction units inside. Clustering algorithm groups objects (action) adapting their features in clusters based on similarities. A clustering algorithm minimizes the intra-class distance (distance between actions composing a cluster) and maximizes the inter-class distance (between the clusters). In our case the clustering algorithm identifies UI containers and computes similarities regarding the minimum distance to reach a common
parent in the task tree \((a_i,a_j)\), this score increases when the actions of the container are hierarchically close together in the task model. Then the score could be computed considering (1) the structure of the Task Model (TM), (2) the user feedback ranking the solution and (3) the cohesion of prediction.

A first variant of the ContentScore is computed according to the TM structure in terms of task hierarchy, relationship and weights. The measure is determined as follows:

\[
\text{ContentScore}_{\text{TM}}(\text{CompoundIU}) = \sum_{t_i,t_j} \left( \text{TaskWeight} / \min(\text{DistanceToReachCommonParent}(t_i,t_j)) \right) \tag{1}
\]

Where:
- \(t_i\): subtasks of a common parent
- \(\text{Taskweight}\): value assigned by considering task type
- \(\text{DistanceToReachCommonParent}(t_i,t_j)\): distance between nodes to reach the common Parent of \(t_i\) and \(t_j\).

AnAlternative score of more complex actions regarding the user behavior prediction (U.B.P.) is defined. The user behavior prediction module is not formed to compute the “container content score” since, by definition, this technique takes the ordering of the actions into account to evaluate the next most probable actions. There is thus an underlying notion of sequence. To meet this requirement and remove the influence of the action ordering, we proceed to simulate all possible history and compute all probabilities of the next task, and then we keep the well-ranked probability for each action. The score of an action is computed as in formula (2). Then the content score is calculated according to the formula (3).

\[
\text{OrderIndepProbability}(\text{CompoundIU}) = \max(S \in \text{SimulatedSequences}) (\prod_{(t,h) \in S} P(t|h)) \tag{2}
\]

\[
\text{ContentScore}_{\text{Prediction}}(\text{CompoundIU}) = \sum_{t} (\text{OrderIndepProbability}(t) \cdot \text{UbPfeatureweight}) \tag{3}
\]

Where:
- \(t\): subtasks of a common parent
- \(h\): history of fulfilling task
- \(\text{UbPfeatureweight}\): weight of user behavior prediction.
An additional relevant metric is the Ordering score, which considers mainly the position of the action inside the container; in consequence it is more restrictive and easier to compute regarding user behavior prediction. We have just to consider the ordering of the actions inside the AUI model and consider its positions. The score is then more restrictive than previous formula. The score of actions given the task model is computed in the following way:

\[
\text{OrderingScore}_{\text{TaskModel}}(\text{Action}) = (i \times \text{task model feature weight} \mid \max_i \text{actions}[0,i].\text{sublistOf}(\text{DFS}(tm))) \tag{4}
\]

Where DFS(tm) generates a sequence of actions by exploring the task tree in an in-depth first search. Thus the more the sequence of actions inside the container respects the order defined by the task model, the more the score increases. As well as the content score, different variants of ‘Ordering Score’ could be computed according to the considered feature. For instance, regarding the Task Model, an AUI could be evaluated for the degree of appropriateness and conformity to the task’s structure regarding both order and hierarchy. The score variant of actions given the user behavior prediction follows the next formula:

\[
\text{OrderingScore}_{\text{Prediction}}(\text{Action}) = \sum_{a|h} \frac{\text{fulfilled actions} + \text{actions}}{P(a|h)} \tag{5}
\]

The main advantage of a clustering algorithm is its scalability for considering new characteristics as new dimensions, besides the possibility of weighting different features in order to discriminate them according to their relevance.

For an easier understanding of the score computation module, a functional analysis is illustrated in Fig. 30. All computed scores are assigned to an object instantiated from the Score class to keep track of the subscore for debugging or analysis purposes. A subscore is a part of the score and it is computed by using a specific feature, such as feedback or behavior prediction.
Chapter 4. Identification of the Abstract User Interface

![Diagram of AUIModel and CalculateScore module]

**Fig. 30 Input-Output calculating score module**

Module 1 describes the calculation of the Tm-content score. This subscore reflects the accuracy of sticking to the hierarchy of the task model. Each subscore is determined according to the weights saved in the context of use. These weights allow tuning the adaptation. For example, we can lower or remove the influence of the user behavior prediction or feedback.

**MODULE 1. CalculateScore: tm-content // stick to task model hierarchy**

- **Input:** AbstractUserInterface.
- **Output:** Score tm-content assigned
- **Var:** costij: integer

```plaintext
for each action do
    costij = tm.calculateDistanceBetweenAction(actions.get(i), actions.get(i+1))
    costji = tm.calculateDistanceBetweenAction(actions.get(i+1), actions.get(i))
    if (costij > costji) then
        costij = costji
    end
    s.tm_content += ((cou.getUser_stick_taskmodel()/costij)/actions.size())
end;
```

In the view of using the user behavior prediction to compute a subscore for the first part, a method called `calculateActionScoreForNextAdaptation()`, illustrated in Module 2, is employed. This method ensures a kind of clustering algorithm with the user behavior prediction tool, which allows us to provide a subscore that is independent of the action position. Indeed, by definition, the user behavior prediction tool takes the position of the actions into account.

The method uses as history the previous actions accomplished by the end-user, and then it updates this history. It does not simulate a single history by adding elements to the same sequence, but it simulates all the histories of previous interaction sessions. Then it selects the next most probable actions after having analyzed all histories. Each subscore is valued according to the
weights saved in the context of use. These weights allow tuning the adaptation and then the best-scored solution is selected and displayed in the next container.

**MODULE 2.** calculateActionScoreForNextAdaptation()

*Input:* previousActions, ProbaOfActions, depth.

*Output:* Action score assigned

*Var:* ArrayList: simulatedPreviousAction, nextMostProbableAction

if(depth > 0) Then

nextMostProbableAction = ubp.getProbabilities(previousActions.subList(previousActions.size() - markovOrder, previousActions.size()))

for (each nextMostProbableAction) do

action = (TaskCSP)tm.getActionByName(nextMostProbableAction.get(i).getKey())
proba = nextMostProbableAction.get(i).getValue() * probaOfPrevActions

action.insertProfitOfAction(proba)

simulatedPreviousAction.add(action.getTaskName())

depth = depth - 1

calculateActionScoreForNextAdaptation(simulatedPreviousAction, proba*0.9, depth)

end end;

**E. M@rina: Methodological View**

In this section, we put forward a simulation of the adaptation process execution (see Fig. 31). We consider a simple task model as input, operating the reification and triggering the adaptation process. The execution of the reification module invokes the triggerAdaptation() method in charge of adapting and generating the first container. With the same trigger, all possible partial AUI are reified and scored via the calculateScore() method detailed in the previous section.

![Fig. 31 The simulation of reification process.](image-url)
Chapter 4. Identification of the Abstract User Interface

The high scored AUI is reified and displayed to the user. Fig. 30 illustrates a partial simulation of the AUI reification for the mentioned task tree. The simulation describes different potential AUI and computes their scores. The highest scored AUI is reified into FUI and displayed for the user. In our case, the second option was displayed (Fig. 31).

4.2.2 M@rina: Support tool

The implemented tool was tested via a case study, and then the results were evaluated regarding quality properties and the ‘Layout Appropriateness’ (LA) criterion.

A. Case study: Money transfer

We considered a money transfer case study, which consists in the money transfer between two accounts. To make a money transfer, the user must provide the beneficiary of the transfer, the amount of the transfer, and then she chooses the debited bank accounts. The choice of the bank account is enabled by specifying the amount of the transfer and if the balances of the user accounts are high enough. The user may optionally provide a comment. Finally, a summary of the transfer is displayed for the user before the validation of the transfer.

B. Generated UIs

The prototype is implemented in Java and provided with a simple module for final user interface (FUI) reification in order to show results. The task tree of Fig. 32 defines the transfer operation steps and the original interface, generated during the first session before gathering the user feedbacks. Then we consider a list of executed sequences throughout diverse sessions. Afterwards, a new interface version is shown according to the new adaptation setting. The system is expected to monitor all accomplished sequences, and implicitly gather additional contextual data affording the user behavior prediction. We execute different sequences to perform both classic and IBAN transfers by changing the order of filling data. Then new adapted UI versions are reified in accordance with the partial reification concept depicted above.

Partial reification requires additional data, specific to the level of abstraction. These data belong to the implementation of the adaptation architecture and they are not needed in case of whole interfaces reification, thus they are not specified in the abstract class AdaptingInterface. These data are obtained a specialized class implementing the methods needed. The class is a
particular architecture adaptation extending \texttt{AdaptingInterfaceArchII}. This specialized class will receive the context of use and the task model initialized by the “main” method.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{image}
\caption{Fig. 32 a) The MoneyTransfer Task model. b) First autonomous adapted UI: (user-feedback independent).}
\end{figure}

The constructor will initialize the reification classes: \texttt{ReifiTMtoAUIM} and \texttt{ReifiAUIMtoFUIM}. It will also initialize the \texttt{UserActionPrediction} tool to use it in the computing of the AUI scores. The \texttt{start()} method allows to trigger an adaptation in order to display the first container to the user. The \texttt{calculateScore()}, \texttt{triggerAdaptation()} and \texttt{“triggerEvaluation()”} are inherited from the parent class \texttt{AdaptingInterface}. The \texttt{triggerAdaptation()} method might be invoked by two different ways. Firstly when the user clicks navigate to the next container, secondarily when she explicitly requests adaptation and the control value turns out to be “true”. Depending on the value of “control”, the \texttt{triggerAdaptation()} method reacts differently. If “control” is true, then the \texttt{triggerAdaptation()} method displays controllability features to the user.
Chapter 4. Identification of the Abstract User Interface

a. Scenario 1

Fig. 33 a) Reification classes Scenario accomplishing a classic transfer. c) Scenario invoking the controllability feature.
By analyzing the results, we found that based on the monitoring of a previous user intervention, the interface is optimized in terms of arranging interaction units. So tasks are ordered according to their probability derived from a task succession during preceding sessions and user choices. For instance Fig. 32 a, b and c presents different versions of reified interfaces considering user feedback and the indicated transfer type. Diverse scenarios are suggested; we note that the initial container is the same for both. However, it is different from the first interface in Fig. 32.b since it is adapted by considering gathered feedbacks during the previous session. Likewise a notable difference could be recognized between proposed scenarios.

In the fig. 33, we display different proposed scenarios for a money transfer. Moreover we underscore the dissimilarities of task arrangement in accordance with the specified transfer types. The first scenario exemplifies a classic transfer (Fig. 33.a) then the IBAN transfer scenario is demonstrated in the second one (Fig. 33.c). Regarding the systems quality criteria, resulted interfaces fill several usability and quality requirements. First functional suitability is maintained along with all operated adaptations. Accordingly the system promotes and demotes tasks (interaction units) based on the user history and user behavior prediction module. On the other hand, optional tasks are delayed until the end of the session without removing them. Which improves the appropriateness for task model and reliability. This reliability is put across the fact that upcoming containers depend on the user choices when using the current one, more than on feedback.

4.2.3 M@rina: Evaluation

In order to improve the validity of the framework we still need an evaluation of the instantiation. Evaluating interfaces is undoubtedly one of the most important validation forms. In this section, we evaluate M@RINA’s layout and technical skills. Two different approaches were processed to evaluate the generated interfaces.

A. Empiric evaluation

The first evaluation addresses the UI layout. We empirically investigated the layout appropriateness (LA) of the generated interfaces, using the factored version of the cost function reported in [Sears 93]. Since adaptation is defined basically at the abstract UI level we focus on the layout and the graphical organization of UI elements. The Layout Appropriateness (LA) metric looks suitable to evaluate and compare generated solutions [Sears 93]. The Layout Appropriateness (LA) idea is to assign a cost for every one.
Chapter 4. Identification of the Abstract User Interface

The cost of a layout is computed by assigning a cost to each sequence of actions and weighting those costs by how frequently each sequence is used. LA has been validated to demonstrate its effectiveness for evaluating interfaces. A specific layout cost is computed via the following formula:

\[ \text{Cost} = \sum_{\text{all transitions}} [\text{transitionFrequency} \times \text{transitionCost}] \]

To evaluate and compare our result we consider the above case study and the first generated layout, then after some user intervention, we generate a new interface with a different layout. The new interface is intended to take the user feedbacks into consideration. Fig. 34 shows the new interface. The comparison consists in performing the same sequence of tasks in both interfaces and comparing costs. The cost computed for both generated layouts (see Fig. 33a and Fig. 33c) was consecutively 46 and 27. According to that result, we note that the cost average decreases in the adapted version of the interface, and then adaptation based on monitoring session optimizes the user interface description. However, we should mention that the cost average would be highest when the user proceeds to the controllability feature for adapting the UI. On the other hand, the cost optimization depends on the Markov model order, however in this case the UI will be stuck to the monitored sequence and prohibits other possible scenarios of use.

Fig. 34 Computing LA cost for the generated interface layout.

Beside the empirical evaluation, we proceeded to a user evaluation of generated UI and adaptations provided during an interaction session. We evaluated two different adaptation modes; user’s initiated and system’s adaptations. The last one considers the adaptive behavior among generated interfaces, and it relies on the factored cost function described in the adaptation section. The user initiated adaptation concerns the adaptable side and consist of the use of the controllability feature to allow them to customize the UI based on their preferences.
B. User experience

According to the Nielsen guide to usability testing; “the strength of the thinking-aloud [Lewis et al. 82; Ericsson et al. 80] method is to show what the users are doing and why they are doing it while they are doing it in order to avoid later rationalizations” [Nielsen et al. 02]. The Think-aloud protocol is a method used to gather data in usability testing in product design and development, in psychology and a range of social sciences (e.g. research, decision making) [Ericsson et al. 80]. Since the early 82, the method was introduced in the HCI and usability fields by Clayton Lewis [Lewis et al. 82].

Think-aloud protocols involve participants thinking aloud as they are performing a set of specified tasks. Users are asked to say whatever they are looking at, thinking, doing, and feeling as they go about their task. A related but slightly different data-gathering method is the talk-aloud protocol. This involves participants only describing their action but not giving explanations. This method is thought to be more objective in that participants merely report how they go about completing a task rather than interpreting or justifying their actions [Ericsson et al. 87]. We decided to elicit verbal data since we believed that this would provide us with objective and accurate information. Considering that the technique was highly useful for gaining an understanding of interactive information behavior.

Experiment: During the study, 12 volunteer subjects (50% women, 50% men), from different fields such as mechanics, biology, and computer sciences, performed the experiment. The average age was 29 years, and all participants were familiar with the computer.

All participants made the experience in the same context. Participants were asked to Talk-aloud whilst undertaking the task (i.e. to verbalize what they are doing as they are doing it). They were not able to ask any questions about the observation, while still undertaking the task. Also, we have found it useful for the participant to show a short (7-13 second) example of thinking aloud when conducting a simple login scenario. This minimizes the time required to introduce the experiment and provides a concrete example of how to think aloud when using an interactive system. Each participant was asked to interact four times with the interface, assumed to be generated and adapted based on his or her preferences, along with audio recordings of the user's thinking aloud while performing tasks. And then we evaluated the interaction according to an audio record reflecting their perspectives on the interactions when using interfaces. This involves participants only describing their actions but not giving explanations. We believe that this method is more objective in
that participants merely report how they go about completing a task rather than interpreting or justifying their actions.

Once we had our recordings, we had a measurement of audio amplitude against time, which isn’t very easy to analyze because of speakers differences and various type of background noises. We wanted to identify where certain efforts were involved when users were talking. We were using this to detect where there was a sound within the audio file. We trimmed silence from records; this function consists in determining a minimum volume (amplitude) threshold and duration (<20 dB for more than 2 seconds) and then simply doing a volume analysis of the waveform looking for areas that meet these criteria. Users were asked to accomplish a “Money transfer” task four times summarizing the two phases of the experiment.

First phase: consist of two interactive sessions accomplishing the full transfer task. It is aimed to verify that the subject could recognize changes in the UI depending on the following sequence when establishing the task. During the first two iterations users did not make an intervention for adaptation. M@RINA is intended to generate user interfaces that are predicted to be the fastest for a particular participant to utilize. The initial hypothesis was that the interface arrangement would vary depending on human errors e.g. displaying the next container without filling all information, and also depending on the order of tasks completion. For a typical scenario where forms are fulfilled in a systematic way, the user will not perceive differences.

Second phase: The second step of the experiment consisted in using explicit feedbacks to manage adaptation. During the third iteration, we asked users to provide an explicit feedback by ranking the UI and/or using the controllability feature when realizing the transfer scenario. The goal was to allow each subject to explicit his preferences for adaptation in order to reduce the interaction load during the last iteration. In this step, we were especially interested in minimizing the period of the final iteration.

C. Results

The trend is clear that the rate of time consummation is lower in an adapted interface. To make this observation statistically significant, the ANOVA ‘One-way analysis of variance’ test is conducted on the value. The table 12. shows the result of the ANOVA test produced by the statistics tool GraphPadPrism.
Chapter 4. Identification of the Abstract User Interface

Fig. 35 a). Average Time-consuming by users. b). Average Time-consuming tasks for different iterations.

We were expecting subjects to be aware of variances when interacting according to the adaptation decisions deployed during the different phases of the experiment (i.e. the workload during second and fourth iterations is lower than the first one). There was a significant effect of adaptations on workload and time consumption at the p < .05 level for the three conditions [F(3, 44) = 3.136 p = 0.0348]. A significant variance between the workload averages of different iterations was shown. The rate of time consumption was significantly different and we can confidently state that workload decreases with adaptations.

In a second phase we analyzed the same set of data by considering three groups of interaction sessions: the first group corresponds to the interaction session with not adapted UI, the second considers the session where adaptations where triggered explicitly, and the last group considers only the session where the adaptations were performed in an implicit way. In a second phase we analyzed the same set of data by considering three groups of interaction sessions: the first group corresponds to the interaction session with not adapted UI, the second considers the session where adaptations where triggered explicitly, and the last group considers only the session where the adaptations were performed in an implicit way.

Table 12 ANOVA test results on average rate of Workload consuming

<table>
<thead>
<tr>
<th>ANOVA table</th>
<th>SS</th>
<th>DF</th>
<th>MS</th>
<th>F (DFn, DFd)</th>
<th>P value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Treatment (between columns)</td>
<td>1142</td>
<td>3</td>
<td>380.5</td>
<td>F (3, 44) = 3.136</td>
<td>P = 0.0348</td>
</tr>
<tr>
<td>Residual (within columns)</td>
<td>5338</td>
<td>44</td>
<td>121.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>6480</td>
<td>47</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

We expected a significant variance between workload during an interactive session with implicitly adapted UI, the session without adaptations and the session where adaptation was invoked explicitly. Similarly, there was a
significant effect of adaptations on workload and time consumption at the p<.05 level for the three conditions [F(2, 49) = 4.378 p = 0.0178].

Table 13 ANOVA test results on average rate of Workload consuming for explicitly and implicitly adapted sessions.

<table>
<thead>
<tr>
<th>ANOVA table</th>
<th>SS</th>
<th>DF</th>
<th>MS</th>
<th>F (DFn, DFd)</th>
<th>P value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Treatment (between columns)</td>
<td>986.7</td>
<td>2</td>
<td>493.4</td>
<td>F (2, 49) = 4.378</td>
<td>P = 0.0178</td>
</tr>
<tr>
<td>Residual (within columns)</td>
<td>5522</td>
<td>49</td>
<td>112.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>6509</td>
<td>51</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Because we have found a statistically significant result in this example, we needed to compute a post hoc test. We selected the Tukey post hoc test. This test is designed to compare each different iteration. We compared the means of three interaction groups (not adapted, implicitly adapted and explicitly adapted) (via one-way ANOVA, multiple comparisons considering iteration 1 as the control group and alpha=0.05. Post hoc comparisons using the Tukey HSD test indicated that the mean score for the implicitly and explicitly adapted interactions was significantly different however no significant difference between adapted and not adapted UI was found.

Table 14. ANOVA test results on average rate of Workload consuming for explicitly and implicitly adapted sessions.

<table>
<thead>
<tr>
<th>Tukey's multiple comparisons test</th>
<th>Mean Diff.</th>
<th>95% CI of diff.</th>
<th>Significant?</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>NotAdapted vs. ImplicitlyAdapted</td>
<td>6.660</td>
<td>-2.175 to 15.50</td>
<td>No</td>
<td>ns</td>
</tr>
<tr>
<td>NotAdapted vs. ExplicitlyAdapted</td>
<td>-3.256</td>
<td>-12.98 to 6.466</td>
<td>No</td>
<td>ns</td>
</tr>
<tr>
<td>ImplicitlyAdapted vs. ExplicitlyAdapted</td>
<td>-9.917</td>
<td>-16.36 to -1.472</td>
<td>Yes</td>
<td>*</td>
</tr>
</tbody>
</table>

D. Discussion

Fig. 35 details the time consumption during the test steps. We can observe that the use of both adaptation approaches reduced the total time spent by users to complete the task. The average time-consumed (workload) is higher for the third iteration since an additional task was executed when adjusting the controllability feature.

The predictive algorithm, used by M@RINA, utilizes the data obtained from a monitoring interaction sequence to predict the users’ next most probable action. The resulting user interface is generated fewer times and succeeding windows are adapted at the same time with only one processing of UI. In the successive version, with predictive features, interface elements are defined according to the monitored user interaction sequence. We note that the Pearson’s ‘r’ for the correlation between iterations was close to 1, which is
called a positive correlation. In our example, our Pearson’s $r$ mean was of 0.6 was positive.

![95% Confidence Intervals (Tukey)](image)

Fig. 36 Tukey: Difference between group means.

We conclude that the amount of time consumed for the iterations is proportional to the subject due to the variety of profiles and skills. In the second phase, we could observe that due to the controllability features the interactive session was longer (Fig. 35.b), however better results are obtained for the following session as more relevant adaptation is applied based on explicitly gathered feedbacks. Greater improvement might be obtained by adjusting the parameter setting that produces the best results when customizing adaptation. Otherwise, it is clearly visible that during the fourth iteration all participants were faster in fulfilling the test (Fig. 36).

We found that there is an opportunity to achieve significant improvement by mixing adaptive and adaptable behaviors during interaction with individuals, and we thus motivated to pursue user interfaces that generate personalized interfaces instead of treating all users in the same way. According to results obtained in the experiments, it can also be concluded that the M@RINA is very effective in identifying user preferences by both adaptation approaches. Further, we claim that allowing users direct manipulation on the UI and the adaptation process improves his/her consciousness and prevents the interaction disturbs.

### 4.3 Conclusion

This chapter presented a first MICAA based instantiation. The tool advances a mixed initiative adaptation for context-aware AUI identification. The proposal considers users feedbacks and ML techniques. It was evaluated empirically and via a user test and showed a significant effectiveness in identifying user preferences by both adaptation approaches.
Chapter 5 User Interface
Widget Selection

The previous chapter showed the operationalization and evaluation of MICAA principles for the AUI identification issue. Likewise, with the goal to validate MICAA practicality, this chapter proposes a second prototype dealing with the widget selection issue. The proposed solution is initiated and studied theoretically and then its feasibility is confirmed through a demonstrative support tool. WiSel consists of a web-based tool for mixed-initiative context-aware selection of concrete interaction units supporting a developed controllability and an enhanced flexibility.

Section 5.1 describes the sub-problem and reviews the related aspects. Section 5.2 presents the mixed initiative widget selection and its support by WiSel. Section 5.3 states the results of pilot users study testing the feasibility of the solution and its impact on usability mainly in term of MICAA-IQS.

5.1 Definitions and analysis of the sub-problem

All the following work is carried out in pursuit of the same goal: support the understanding and the feasibility of MICAA of UI. This section presents a second prototype addressing the contextualization of UI layout. At concrete UI level, adaptation consists of restyling, positioning and arranging concrete interaction units. One of main interaction’s concerns is to define the appropriate interaction unit at a widget granularity level for a given context.

5.1.1 The problem of widget selection

Widgets are reusable visual elements of a graphical user interface, which are designed to support interaction’s input/output flows. One of the key issues of the UI layout’s adaptation is “How to select the right widget for the
appropriate context matching the user task and agreeing with the user preferences and interaction context?”. Different widgets could be defined for the same task depending on their properties and types. Further, other context factors have to be taken into account when defining the UI’s interaction objects such as screen size, platform modality, user’s expertise levels etc. Assuming that a context-aware definition of a widget prevents usability and ergonomic problems, diverse techniques and rules are considered to ensure a context-aware identification of interaction units. The main goal of context-aware widget selection is to comply with visibility and accessibility directives to improve the UI usability level.

The requirement for context-awareness is common and recurring for UI definitions. There is a significant need for further considerations on the growing complexity of interaction systems, the multiplicity of widgets and the context diversity. Quite few works [Puerta et al. 99, Demeure 08, Mitrovich, et al. 04] have been done to improve the context-awareness of widget selection. Typically this task is delegated to UI designers, who have difficult choices to make without guarantee of making the right selection for a given context.

In what follows, we review the literature related to the context-aware widget selection by classifying existing algorithms into two classes: Knowledge-based and user-centered widget selection.

### 5.1.2 Existing solutions

Most of the existing works addressing the adaptation of widget selection deals with automated UI generation. Diverse techniques were evaluated to ensure the mapping from abstract interface specification to a concrete one. The main purpose is to select and display the right interaction unit at the right context for the suited user. Otherwise, an inappropriate selection may lead to significant accessibility and usability issues. Previous works addressed this issue from different perspectives. We classify proposed solutions for widgets selection in two categories: knowledge-based and user-centered approaches.

- Knowledge-based approaches recognize the intelligent adaptive behavior for full automatic widget selection. It started back to 1988. Kleyn, 1988 proposed a formalism that consists of decomposing a dialog into a set of events specified as an and/or graph. The traversal of this graph results in all possible sequences of actions a user might choose to invoke in accomplishing a particular task. Later, a different solution was developed in the context of advancing model-based UIs generation. TRIDENT [Vanderdonckt & Bodart 93] was the initiation of the intelligent systematic
Chapter 5. User Interface Widget Selection

widget selection. SEGUIA [Vanderdonckt et al. 95] implemented an expert system with rule-based inferences for identifying widgets. This approach was advantageous by its simple format and the justification of choices. Nevertheless the flexibility was the major limitation of such approach, further the definitive commitments of system adaptation decision prevent user-centeredness.

- The user-centered approaches focus on the users’ preferences during execution to select the appropriate interaction unit. Assuming that, the user’s involvement is of great importance for the validation of context-aware adaptations, several methods enabled the user’s involvement through different scenarios and at diverse levels of the interface definition. Decision trees, Markov-based models and different ML techniques are used to perform the automatic matching for widget selection with regard to user’s feedbacks [Mitrovic et al. 04, Mitrovic et al. 05, Demeur et al. 08]. Existing systems employed the model-based approaches by introducing abstract models that target different context dimensions. Most of them proceed to the runtime monitoring of the users’ behavior in order to anticipate their future actions and to identify their users’ preferences. Nevertheless gathered user’s feedback may be deployed to endorse the system decision during future sessions.

Doing “the right adaptation” requires that it be right in the current context for a given user’s need and expectations. The frustration (when user is not satisfied with the UI) and the disruptions (when user is prevented from achieving his/her tasks.) are often due to some poor adaptation decision and/or an unreliable system decisions. Such decision could be correct given the software’s assumptions, but wrong and disturbing for the users. The principal weaknesses are that the adaptations do not meet the users expectations in a given context, the users do not have the full control over the interaction processes, and in view of that adaptations are not transparent, predictable neither controllable for users.

5.1.3 Motivations for context-aware widget selection

Quite few studies in the literature investigated different shortcomings of interface adaptations and called to consider an intermediate adaptivity level [Kaber et al. 04]. The main requirement is to overcome the transparency and controllability deficiencies while keeping advantages of automatic adaptivity [Dessart et al. 11]. Nevertheless, with respect to previous contributions, we believe that the requirement for context-aware runtime adaptations must take up three particular challenges: transparency, predictability and controllability.
Chapter 5. User Interface Widget Selection

The rationale for improving interaction is putting forward user at the heart of interaction and adaptation process. The challenge is to bridge the gap between user’s needs and adaptations. We are aiming for greater synergies between UI and users improving the reliability of adaptation and streamline the interaction [Mezhoudi et al. 15 B]. A more detailed explanation of synergetic interaction’s requirements, motivations and supports is presented in the section 3.3 (MICAA-IQS).

Controllability is considered as the primary concern in mixed-initiative adaptation: controllability should be balanced with predictability [Findlater et al. 04, Kaber et al. 04]. Accordingly, end-users should be integrated in the adaptation loop [Evers et al. 12] through both implicit and explicit feedbacks. Each type of intervention (explicit, implicit) contributes to the interaction quality. The implicit control allows the understanding of user’s intents. The explicit control provides users the capability to change UIs according to their preferences and offers the system more effective feedback.

Transparency was addressed from different perspectives, [Dessart et al. 11] target the visualization. Further, the transparency was addressed otherwise by self-Explanatory UI, which have the capacity to provide the end-user with information about its purpose, structure and design. Both visions are complementary, and when combined, create the best creative solutions increasing transparency and improving trust between users and systems.

Predictability was stressed as a crucial evaluation criterion for adaptive UI [Gajos et al. 08], however most of works do not address it explicitly through a focused solution. We consider predictability as the key issue; this requirement is mainly supported by reliable adaptation. Such reliability could be achieved via two different ways: In the anticipatory stage via a post analysis of the potential situation of interaction. In post-interaction, to comprehend and recognize involved features that influence the interaction [Mezhoudi et al. 15C].

The analysis of the requirement for context-aware widgets’ selection challenges the need for a more user-centered, rational adaptation mixing both reactive and proactive behaviors:

- The Reactive UI behavior outlines the classical adaptation practices (designers’ rules). Both adaptive and adaptable instantiations capitalize on a set of predefined adaptation’s rules triggered on the user request and/or contextual event. Predefined adaptations consider designers knowledge about usability and context support. However there is no consideration for unusual situations and/or inexperienced context factors. The main shortcoming is that adaptation’s rules are mostly defined at design time and do not evolve across time. For a given context, the UI has enormous
variety of possibilities for how to react/adapt to the context and the user’s inputs. Usually, they just pick one of those possibilities, more or less arbitrarily. These shortcomings call upon new adaptation modes, inferring up to date knowledge’s and above all validating system’s adaptation decisions processed during runtime.

- The proactive behavior (Self-learning) conveys adaptations based on the system’s inferences and decisions making. It consists on building UIs around the notion of advice as the primary means of communication. The computer can both give advice to the user (assistance) and take advice from the user (learning). Several advanced algorithms and techniques are intended at a more efficient decision making process and more skillful cognitive layer. Systems’ decisions should be progressed at runtime and have to consider both the functional and/or interface levels. For this purpose evolving the interaction style from a command structure to a more flexible and collaborative stance is a priority.

Based on the above, runtime adaptations still carries potential for growth. There are three key factors to be considered when performing adaptation: prediction, proactiveness and controllability. Adaptations decisions should as well provide all users high satisfaction levels. We believe that considering MICAA approach in respect of MICAA-IQS can contribute to the improvement of the user interaction quality, the understanding of system’s actions and the acceptance for the system’s adaptation.

### 5.2 The proposed solution: WiSEl

Considering the above-mentioned requirements and challenges, we put forward a MICAA based solution for adapting UIs at concrete level and specifically the interaction units. WiSel is aimed at providing users a mixed-initiative context-aware approach for adapting interfaces. Following the MICAA’s principals, WISEL’s support a mix of user and system control on adaptation at runtime with regard to the users’ context and preferences [Mezhoudi et al. 15C]. This allows a more precise assessment of the context of use and a better understanding of users’ needs. Fig. 37 depicts the characterization of WiSel adaptation approach with regard to the MICA-Str.
Chapter 5. User Interface Widget Selection

The implementation conveys the WiSel adaptation process considering the user as the main parameter for deciding and triggering adaptations (To what?). User preferences are acquired by means of both explicit and implicit feedbacks during interaction (When?). Adaptations are processed and applied at runtime through a particular feature for context-aware widget selection and a context-aware widget recommendation. The context-aware widget selection starts with score-based identification to end up with controlled/approved widget selection (Where?). The functional part of the system (How? What? Why?) capitalizes on context-aware similarities and scores-based adaptations. The score based adaptation support being initiated by mixed-initiative (user and systems) and adjust the UI at runtime. The context-awareness of adaptation decisions derives from an adaptation manager to ensure a context-aware selection of interaction units and a context-aware recommendation at runtime (Fig. 37).

As far as we know, this approach is unique, in allowing users direct explicit adaptation of interaction units without interrupting the main task (filling out form). On the other hand such interactions are interpreted by the system in two different ways. First users’ interventions are considered as a main entry to create the user’s profile. Further users feedbacks are involved to handle adaptations decisions through a score-based technique. We believe that those reflections are advantageous for adaptation context-awareness and help to encourage users to invest in customizing the UI.

Fig. 37 MICAA based CUI selection: WiSel instantiation
Furthermore, WiSel approach supports the explicit feedbacks during runtime; it authorizes the user a full control on the interface’s interactive elements while assisting them by recommending potential choices. The user-controlled adaptation supports the continuity of interaction and prevents user’s disruption. This adaptation is more consistent with the whole interactive session. However, the user’s interventions should be handled with care to avoid overloading users and/or confusing their interaction.

WiSel assume that the consideration of hybrid strategy for selecting widgets allows the systems to take advantages of a vast number of adaptations’ potentials. Two methods are recognized, with diverse policies and perceptions adaptivity and adaptability. This mixture is aimed at enhancing a context-aware widget’s selection process through overlapping different prospects and improving the decision-making (Fig. 38). The mixed-initiative adaptation is mainly designed to maintain equilibrium between the end-users control (direct manipulation and feedbacks) and the systems adaptation (recommendations and adaptive behavior).

**Knowledge-based adaptations:** (Designer-based adaptation) At first levels the system puts forward a typical adaptation approach that we call knowledge-based (design-time) approach. In this case, adaptations are defined regarding a set of pre-established guidelines and designers instructions. Typically adaptations rules are based on usability and ergonomic guidelines. A set of adaptation patterns are defined matching expected context requirement, this technique is particularly advantageous in the design phase to drive decisions that may improve the overall quality of the interface. By analyzing models, developers may anticipate imperfections that would disrupt users and decrease usability. However design-time assumptions may change during execution. Accordingly the static aspect of this approach deprives the consideration of the new unexpected context, the users requirements and prevents the computing of appropriate adaptation patterns. To overcome such shortcoming end-users were involved in the adaptation process. This fact was supposed to enhance the matching between knowledge-based widget selection and context requirements.

**User-based adaptation:** During execution, the changing end-users supplies could be identified through the interaction sessions therefore taking the user into account for updating design time adaptation models is claimed. WiSel approach supports user-centered adaptations via different consideration levels; first by considering the user expertise level to define the interaction model and then by granting users the full control on adaptation, by inferring
knowledge based on the users’ intervention to revise the existing adaptation patterns.

Both implicit and explicit users’ feedbacks are supported. Adaptation’s decisions rely on runtime gathered feedbacks. Further user-centered adaptations are established by considering the users’ profiles defining the user’s expertise level to identify the interaction model. WiSel considers three expertise levels during adaptations processing (novice, medium and advanced). From the system side, the support of explicit feedbacks instructs a proactive behavior for adaptation decision-making and supports an “intelligent” user-centered widget selection. On the other side, WiSel authorizes the user with a full control on the interface while assisting them by recommending appropriate interaction objects. Allowing users the full control is intended to be coherent with the whole interaction, to prevent overloading users and to eliminate their disruptions.

![Diagram of widget selection process](image)

**Fig. 38** Widget selection process: Three adaptations directed support the adaptation process, the designer knowledge, the user direct manipulations and the advanced logic based on the scoring function.

Fig. 38 outlines WiSel’s widget selection process. During the interaction session, each user can manage adaptation and change the interface’s widget according to his or her preference without disrupting the principal task. We consider explicit users interventions to allow interface personalization and to draw the user profiles and preferences (requirements). Those information are considered as the input for the score-based adaptation and the recommendations module. Users involvement is assumed to provide a valuable
appraisal for adaptation (Refine adaptation patterns). It highlights the key strengths and limitations of adaptations approach. Further, providing users with direct control permits to decide personalization and improves the context-awareness, user-centeredness and effectiveness of adaptations.

*Score-based adaptation:* Given that customization can be an overloading task for end users, a widget recommendation mechanism is provided. WiSel assumes that full user control may end up being more confusing than helpful to end-users. To prevent such difficulty, users are provided guidance for adaptation choices besides multiple level of control with regards to their expertise. In case of knowledgeable users, the recommendation may be considered as a simple visualization of the potential adaptation. For inexperienced users, the recommendations’ list provides guidance, for example, in case of choice widget selection, the list of interaction units that can undertake the user task is defined with regard to other users’ choices and the widget appropriateness for the task. Recommended widgets are sorted according to the scoring system. Further, users can delegate the adaptation task to the system by considering the displayed (promoted) widgets, which is already supposed to be appropriate to the context of interaction.

We believe that WiSel approach can improve the accuracy of adaptation decisions. The fact that only interventions of experienced users are considered, allows to develop up-to-date adaptive behavior adjusting adaptations with regards to real and trusted aspects.

### 5.2.1 Algorithms

In this section, we present WiSel methodological instantiations. Assuming that WiSel flexibility capitalizes on a proactive and reactive management of adaptation, diverse gadgets are involved in the adaptation process. WiSel focused mainly on involving users in the adaptation decision-making process and this support is managed in different ways. Assuming that supervised recurrent upgrades of adaptation with regard to user feedbacks can improve interaction. Gathered users’ feedback at runtime seems to be more accurate for adaptation. We proceed to direct manipulation for customizing UIs according to their users’ preferences besides the user’s profile.

At the most fundamental level, the user profile outlines a set of criteria that defines the user roles and expertise level. WiSel’s profiles consider the user’s expertise level but focuses on the main part on user previous interaction pattern. The expertise level is considered to define the interaction mode and to decide the level of control granted to users. This distinction is aimed to provide the system more reliability and avoid incorrect user’s arbitrary choices.
The second factor of the user-profile is established with regard to their interactions. It consists of combining a set of vectors establishing user’s selected inputs for previous displayed interfaces (forms). Based on those profile vectors and expertise level WiSel establishes a similarity matrix and compute recommendations. The system of learning-based adaptations is based on analyzing gathered feedbacks. Feedbacks are intended to serve in promoting/demoting adaptation choices. Consequently it assigns a rule’s priority supporting adaptation decision and allowing to resolve conflicting situations. WiSel implements a rules learner engine based on a supervised learning approach with a training phase. It is aimed at upgrading adaptations by inferring novel acquired knowledge <Refine Adaptation Pattern> (Fig. 38). Adaptation decisions are handled by means of the following accomplishments: 

(1) Executing pre-existed adaptation rules, which serve as away of a training set to 

(2) Detect a pattern of user behavior throughout his/her interaction and feedbacks. 

(3) Besides, coming up with statistics and (promote/demote) ranking for learning adaptation rules. 

At the first adaptation phase, the system puts forward a typical adaptation approach that we called knowledge-based approach. In this case, adaptation is based on established guidelines and the designers’ adaptations intended to be upgraded with new acquired data. In our tool, the knowledge-based feature outlines the reactive adaptation behavior. It is supported by a set of predefined adaptations allowing the mapping between tasks types and interface elements. This set of rules outlines the kernel of the adaptation engine allowing the generation of “context-aware” interfaces. Despite the inaccuracy of preexisting rules, defined adaptations are sufficient to feed previously displayed interfaces. The usefulness of established adaptation rules is also to provide a system with the basic knowledge and guidance to build on those rules during the learning phase. The support of user’s intervention is aimed to enhance adaptation learning of new knowledge and acquaintances. Diverse advanced algorithms assist the reasoning and support the interface proactivity. The goal is to produce clearer and more accurate deductions regarding learned knowledge. For instance end-user selection of a widget endorses its appropriateness for the task within the current context. 

Fig. 39 denotes the class model of WiSel recommendation. It consists of an ElementaryInteractionObjet class describing an abstract description of an interface’s elements. Such elements are related to task-based specifications allowing the refinement of appropriate concrete interaction objects.
“Widget” class is the specialization of \texttt{ElementaryInteractionObjet} and defines concrete interactors. Through our instantiation of MICAA, the default widget is selected and associated with each \texttt{ElementaryInteractionObjet}, rather the recommendation list allowing users to adjust their final interface within an assisted self-directed adaptation approach. The \texttt{RecommendationList} ensures the assistance for adaptation. It consists of a set of scored widgets guiding the definition of final interactive units.

Several options are considered for the scoring function permitting the adjustment of involved parameters. We assume that providing flexibility handles the lacks of understanding of adaptation decisions. The configuration of scoring parameters explains the rationale underlying the adaptive behavior as well as recommendations and enhances the predictability of adaptation approach. \texttt{WiSel} provide designers a manageable flexible scoring function allowing the definition and configuration of the involved parameters. The score configuration consists of a first phase to select a list of considerations to be involved during recommendations such as the user’s experience and the admin’s supports for some choices etc.

Then three scores were defined to guide the recommendation:

- \textit{Score of change} (SC): This score defines the plus granted to a widget after being selected by a user or a designer for a specified interaction unit. Such score allows the recommendation the promoting/demoting of widgets with regards to users and experts.

- \textit{Score of unchanged} (SU): these values will define the interest accorded to the system choice in term of rewarding a well-behaved recommendation within a reinforcement-learning paradigm.

- Both designer score (SA) and a global score (SG) exhibition depends on the previously selected options. They present respectively the relevance
Chapter 5. User Interface Widget Selection

accorded to a widget when it is recommended by the designer (admin), and the significance accorded to the frequency of a widget selection.

Four potential score formulations were determined with different considerations. Table 15 determines different potential formula.

The basic score considers mainly the score of changes and the score of un-changes. At this level the widget selection depends only on the previous behavioral changes. Where:

- \( P \): the number of times when the widget is selected without being displayed by default (\( W_{\text{Selected}} = W_{\text{Default}} \)).
- \( D \): is the number of times where the selected widget is already displayed. (\( W_{\text{Selected}} \neq W_{\text{Default}} \)).

\[
\text{Score (Widget)} = P \times SC + D \times SU
\]

The most accurate score considers all above factors, both popularity of widgets that represents end user choices and the expert recommendation acquired by designers choices contribute to the assessment of appropriateness of widget to the current context.

\[
\text{Score (Widget)} = P \times SC + D \times SU + T \times SG + f(w,SA)
\]

\( f(w,SA) \) allows to determine if the selected widget matches the designer recommendation. \( T \) is the total number of widget selection.

<table>
<thead>
<tr>
<th>Admin</th>
<th>Global</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>No</td>
<td>( C \times SC + U \times SU + G \times SG )</td>
</tr>
<tr>
<td>No</td>
<td>Yes</td>
<td>( C \times SC + U \times SU )</td>
</tr>
<tr>
<td>Yes</td>
<td>No</td>
<td>( C \times SC + U \times SU + f(w,SA) )</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>( C \times SC + U \times SU + G \times SG + f(w,SA) )</td>
</tr>
</tbody>
</table>

On the other hand, the reactive WiSel behavior offers a significant flexibility allowing users to personalize widgets while being assisted by intelligent recommendations. Recommendations reduce the uncertainty of users and provide user valuable guidance for suitable adaptation. WiSel proceeds to recommendation based ‘K Nearest Neighbors’ (Knn) algorithm to support this guidance. In order to provide designers as well as end-users the ability to manage the interface according to their preference while being assisted and guided by a recommendation feature. A K-nearest-neighbors (Knn) based recommendation algorithm is implemented following the Amazon
Chapter 5. User Interface Widget Selection

recommendation system [Linden et al. 03]. Different features are involved; first the feedback, the user’s profiles, similarity scores and similarity matrix.

Module: KNN based widget recommendation

**Input:** D: the set of K training profiles,

- p : the test user profile, W: the set of widget

**Output:** score prediction (p,j)

Foreach user p ∈ D do

- Compute d(p,p'), the distance between users
- Compute A: Frequency matrix, aij: (user, widget)

End

Select D_k ⊆ D, the set (neighborhood) of k closest training object for p;

ScorePrediction(p,j) = \frac{\sum_{p=1}^{k} sim(i,p)a_{pj}}{\sum_{p=1}^{k} sim(i,p)}

The whole recommendation scoring factors as well as similarity formula permit a flexible adjustment of parameters and allow users to manage the similarity. Four scores are considered for adjusting recommendations:

- Score of **null-compatibility** when (widget) has not been used by compared users’ profiles;
- Score of **undetermined compatibility** when one of user already used the widget while the second did not test it yet;
- Score of **compatibility** in case both users consider the same widget choice;
- Score of **incompatibility** in the case of different users’ choices.

As well as scores, we provide users a flexible recommendation algorithm allowing them to define their appropriate configuration. First the users can “reset” their user profile vectors. Moreover the interface allows users to select among a set of similarity-index-formula such as Cosine similarity, Hamming distance, Jaccard index etc, to determine the number of users involved in the Knn computation. Once all recommendation settings are adjusted, the recommendation algorithm is automatically operated within final interfaces. In what follow, we depict some interfaces of WiSel support tool putting forward above expressed details.

5.2.2 WiSel: Support tool

WiSel’s implementation consists of a web-based environment extending the open platform for web content management Drupal; this platform is used for creating online forms for information’s systems.
As we previously explained, WiSel adaptation operates two different approaches realized via two interaction modes. Further all users’ interaction flows are cogitated for adaptation. WiSel tools build a user’s profile based on implicit and explicit feedback. This profile is recognized as an indicator of user preferences. A log file is established for each user (Fig. 40 ①). It reports the interaction sessions and allows the identification of users’ interaction patterns, their preferences and their profiles. Gathered information are implied to compute the appropriate interaction object for the user in a given context. Also those information allow the system to define the “user’s profile vectors” ②.

User interaction details award an indicator determining users’ satisfaction levels and/or user satisfaction about current adaptation decisions. Moreover, users’ interventions during “the editing mode” enable a runtime customization of UI to user preferences and/or to an unexpected situation. WiSel capitalizes on recent user feedbacks as an input for the “proactive-behavior” involving users feedbacks within the decision-making process. A widget recommender is given in order to facilitate the understanding and the use of the customization mode. Recommendations are computed with the object of offsetting the workload associated to deciding adaptation.
Fig. 41 A. Widget personalization of ‘Specify color’ interactive tasks B. Form for delineating the scoring functions

Fig. 41 A depicts the editing mode for personalizing a ‘Specify color’ interactive tasks. The users are allowed to alter to the editing mode without interrupting the interaction.

![Form for delineating the scoring functions](image)

Fig. 42 The scoring function configuration
Chapter 5. User Interface Widget Selection

Editing consists of displaying a recommendation list of potential widgets. Displayed recommendations result from above detailed scoring functions. WiSel offers a typical graphical interface for online forms. And enables a fast and easy adjustment of adaptation methods (Fig. 41.B). We believe that allowing users to customize the scoring function improves their understanding of the adaptation logic and the reliability of each specific formula. The score value is defined via a different combination of parameters and selected options. Mainly three reflections conduct those scores (Fig. 38): the consideration of designer’s choices, consideration of global score and the consideration of user’s expertise. The user expertise level is intended to guide the learning process and distinguishes valuable UF.

WiSel aims to contribute to significantly user-centered context-awareness, with significant support for controllability. This support manifests in different ways; adaptation’s control (editing mode) and adaptation’s management. This allows the user to control, to adjust and to weight adaptation parameters through a configuration form. Fig. 41, Fig. 42 outlines WiSel interfaces for defining the scoring function and adjusting its parameters. Once the configuration of scoring function of recommendation are completed and considered parameters are tuned, the recommender is offered where and when the user needs. During this study the learning feature is not tested because each user make a single reservation task. The defined form outlines 17 interactions’ tasks allowing users to provide their information and preferences for renting cars (Fig. 44).

Fig. 43 Car reservation task tree
Chapter 5. User Interface Widget Selection

The task model breaks down the whole task of renting a car into simple interactive user tasks. These tasks are fulfilled via an interactive system allowing end users to specify information regarding their contact details and preferences for the rental (including: period, car type, color, model, engine). The variety of interaction objects allows a wide variety of recommendations and decisions. The first displayed interface is based on a system prediction of best widget for each user this recommendation is based on the similarity studies and Knn algorithm based on a similarity matrix between users.

Fig. 44 Widget personalizations of ‘Specify color’ interactive tasks.
The Fig. 44 shows an example of changing widget during an interactive session. We consider the scenario for changing the widget of choosing color. The initial interface uses a color textfield widget, where users are required to write the RGB codes corresponding to their preferences (RGB code is a numerical value allowing to construct all the colors from the combination of the Red, Green and Blue). Switching to a visual color picker is more comfortable because the users do not have to remember codes and do not have to enter it every time. The widget will be shown when the mouse passes over the recommendation list. The user can then select his/her preferred widget and continue his/her reservation.

5.2.1 WiSel: Validation

During the pilot study, participants were invited to personalize their interaction objects regarding their preferences while fulfilling the car reservation form. Initial observation of this pilot study indicates that users invest for the personalization differently regarding the data type and interaction complexity. However, all participants had an overall positive attitude toward personalizing the interface according to their preferences during use without interrupting their current tasks.

All users had tried personalization at least once. During the test 935 interactions were accomplished with different complexity levels. Almost 30% of interactions were aimed at personalizing and changing the displayed widgets. Users conserved recommended widget for 740 interaction units. The table 17 reports those preliminary results. It outlines results of observing 9 selected interactions units designed for different pieces of data in the car rental case study.

First we outline potential widgets that match the data type and that are appropriate for the interaction unit, and then we present a follow-up to obtain the accord between user’s choices and recommendations during the pilot study. The matrix associated graph presents the decision similarity between users and systems for associating a specific widget for an interaction unit. The progression of distances rates presents an indicator on the system learnability. A distinct matrix with appropriate values is treated for each question for each fold (interaction units). Afterwards, we summarize all values for each index and each matrix calculated for each field to make a conclusion about the WiSel adaptation performances.

In some cases one widget totally dominates a specific data type (last name, category, maximum). However, in case of complex interactions more factors need to be considered for instance number of options, multiple/simple
choices. We noted that most personalization were made for complex interaction unit. The confusion degrees between system elected widget and users choices are weak compared to simple interactions.

Table 16 Confusion matrix of “gender selection” interactive task

<table>
<thead>
<tr>
<th>Chosen/Predicted</th>
<th>Radio bottom</th>
<th>Select list</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Radio bottom</td>
<td>21</td>
<td>15</td>
<td>65.5%</td>
</tr>
<tr>
<td>Select list</td>
<td>14</td>
<td>5</td>
<td>47.3%</td>
</tr>
<tr>
<td>Total</td>
<td>35</td>
<td>20</td>
<td>47.3%</td>
</tr>
</tbody>
</table>

We assume that WiSel ‘s recommendation context-awareness will be improved after the training phase. During this phase the system realizes self-learning by capitalizing on the recognition of users behavioral patterns and similarity. Further, monitoring the interaction scenario will improve both transparency and predictability of adaptations for users. This is supported and demonstrated by the degree of involvement of users in the personalization tasks (controllability feature).

The graphical representation in table 17 is intended to draw the evolution of system decisions. Improvement capitalizes on measuring the distances between predictions and user choices during the personalization process. This study is only an initial step, but the approach appears promising to be extended and generalized. Engaging users for adapting the interface is difficult, however the majority of participants invest voluntary for personalizing their interfaces. To sum it up, we could say that the approach can be fully operational. The first pilot test was satisfactory and in spite of the individual differences, it has been possible to detect similar attitudes toward the controllability feature. However privacy issues is still not supported by the approach design. Future experiments should include additional means of validating usability tests, user investment and user satisfaction.

5.2.2 Discussion

In his current state WiSel support crowdsourcing for adaptation. We share a common definition with CrowdAdapt [Nebelig et al.13]. The key idea consist of allowing end-users to adapt the interface to their specific use context if it is insufficiently supported by the current design. The objective is to involve user in defining new interface layout that can be displayed for other user.
Table 17 Pilot user study results.

<table>
<thead>
<tr>
<th>Data Unit</th>
<th>Confusion Matrix</th>
<th>Graphical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Last name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gender</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Birth Date</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Country</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Period location</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Category</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maximum</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Color</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Option</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Pilot user study results.
5.1 Conclusion

Specific knowledge and know-how for cross-border cooperation are supposed to reduce the needed customization charge. As we promote sharing and reuse within the user’s community.

Our approach is distinguished by learning the right skills and developing effective strategies for adaptation through making a supervised crowdsourcing. We believe that considering users’ profile and expertise level is important to facilitate the identification of good adaptation practice in a given and to better assess the practical impact of user decision through post evaluation. Furthermore, tracking the data over a period of time reinforces its validity and enables the system’s adaptation engine to anticipate potential context-aware widget. However, developing crowd-sourced adaptation were not in the scope of this thesis. Our main focus was to investigate user interface adaptation based on user feedbacks and machine learning.

This chapter presented a second instantiation of MICAA for concrete UIs. The associated tool advances flexibility, user-centeredness, and controllability for runtime widget adaptation. Adaptations are triggered by both users and UIs at runtime for adjusting interaction units. The proposal used KNN algorithm for User Action Prediction and considers different users’ feedbacks. The user pilot study has demonstrated the feasibility of WiSel. Next chapter is intended to investigate MICAA differently in view of online news recommendations.
Chapter 6 Context-aware Recommendation

This chapter’s aim is to validate the MICAA framework from a different perspective. We focus on improving context-aware recommendation of news based on user feedback. The objective is to optimize the user experience with online journals by recommending items that meet user preferences. In particular we discussed the importance of context-awareness of social recommendation and user involvement through a mixed-initiative approach.

We analyze existing works on context-aware recommendation systems, and we provide solutions that may offer more accurate recommender for online news journals with regard to the MICAA framework by supporting a context-aware feedback-based recommendation.

6.1 Definitions and analysis of the sub-problem

6.1.1 Context-aware recommendation issue

The incredible volume of online information overloads users. Obviously, to find the relevant information or to perform a desired function is a challenge given the growing volume and complexity of digital information, and it will continue to do so in the future. Recommendation is a valuable solution for different fields, most of Internet users have to come across a recommender system i.e., friends’ recommendation on social networks, products recommendation, news recommendation, etc.

Existing recommendation approaches contribute to the improvement of information accessibility but still require further refinement. This chapter will consider specifically recommendation of news article on online journal. Despite the meaningful improvement of recommender systems in general,
Chapter 6. Context-aware Recommendation

there are still challenges that restrain the effectiveness of existing solutions, such as context-awareness and changing user interest. The common goal of recommenders is to improve the access to the relevant information for readers and encouraging data consumption. Originally most recommenders have been evaluated and ranked on their Accuracy, which denote their prediction power; their ability to accurately predict the user’s choices.

Nevertheless predicting and recommending the most relevant news differ in several ways from other recognized types of recommender systems such as for music and films. First given that news articles significance covers a very short period, recommendation have to be adapted and updated continuously. Moreover recommended news must meet specific rigorous selection criteria to ensure coverage, timeliness and prevent convergence. These specificities cannot be set aside and can be summarized as:

(i) **Recency/Freshness** is the quality of information of being an actuality, it is sometimes more important than relevance. This criterion is significant for the relevance of a recommendation and its completeness.

(ii) **Popularity** in the context of Search Engine Optimization (SEO), the popularity of a website is the number and quality of external hyperlinks to this site or inbound links. The popularity of a website is a very important element in the quality of SEO and thus positioning on the results of search engines. In our case, we define the measure for an article in a social context. So an article is "equipped" with a social popularity whose value is scalable, fragmented and explicit.

(iii) **Relevance** represent the need for a significance and variety in recommended news, in the other hand, similarity between articles does not necessarily mean they are related–unrelated despite sharing same key words.

Recommending relevant news with regard to user preferences is not an exact science and faces several major constraints. (1) News readers might have special preferences on particular and unrelated trendy news that might be of interest. (2) The relevance of news depends on the context of reading (time, place, device etc.). (3) Recommending news entails special requirements regarding scalability: a considerable amount of novel news comes and must be phased.

There exist two categories of classic recommendation methods: collaborative filtering and content-based methods [Bobadilla et al. 13]. The majority of derived techniques to recommend focus on recommending the most relevant items to individual users. In the recent past, context-awareness has emerged as a popular approach for several disciplines. Recommending news with regard to the context of reading was promising to improve the
quality of recommendations. Context-aware recommender systems produce more relevant recommendations by adapting them to the specific contextual situation of the user [Adomavicius et al. 11]. Several approaches use contextual information in different ways. The next section provides an overview of incorporating contextual information in the recommendation process.

### 6.1.2 Existing solutions on context-aware recommendation

In this section, we do not review existing recommendation algorithm, we focus on context-awareness of recommendation. Researchers in [Adomavicius et al. 11] put forward a general high-level framework for characterizing the multifaceted topic of contextual information in recommender systems.

According to [Adomavicius et al. 11] three paradigms for integrating contextual factors could be identified:

1) **Contextual pre-filtering** method, where a context-aware data selection is accomplished before applying a standard recommender system;

2) **Context post-filtering** method, where classic recommendation approaches are accomplished and then results are managed considering context; and

3) **Contextual modeling method**, where context is integrated directly into the predicting model.

Fig. 45 Paradigms for incorporating context RS [Adomavicius et al. 11]
Chapter 6. Context-aware Recommendation

The approach proposed in this work belongs to the last category of contextual modeling. In contrast to pre- and post-filtering methods, the context-aware modeling uses context data along with predictions [Adomavicius et al. 11].

6.1.3 Motivations and challenges of context-aware recommendation

The need for context-awareness is evident and already discussed in previous works. Assuming that news recommenders aim to provide “the ‘right’ information, at the ‘right’ time, in the ‘right’ place, in the ‘right’ way, to the ‘right’ person, different context specification could be considered.

The depiction of context and the way of gathering data influence the accuracy of recommendations. The significant diversity in the context of use raises the challenge to meet changing user interest. Thus there is a need for involving users to confirm the validity of context-aware recommendation and contribute their personalization. We believe that feedback-driven recommendations will improve context-awareness and confirm this. [Ozgobek et al. 14] represents an extensive study of news recommendation challenges. Table 18 summarizes most challenging requirements. A wide-ranging list of news’ recommendation challenges is depicted in Appendix D.

Table 18. Keys challenges for news recommendation based on [Ozgobek et al. 14]

<table>
<thead>
<tr>
<th>Challenge</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cold-start</td>
<td>First-rater, ramp up, early rater: It denotes the problem that the system cannot recommend new items if they do not have any clicks from other users.</td>
</tr>
<tr>
<td>Data sparsity</td>
<td>The possibility of data sparsity increases if the number of columns or rows is much higher than the other (articles/readers).</td>
</tr>
<tr>
<td>Recency</td>
<td>Most of the users want to read fresh news instead of old dated articles. So the importance of news items decreases in time.</td>
</tr>
<tr>
<td>User feedback</td>
<td>The system should be able to collect implicit feedbacks effectively while protecting the user privacy.</td>
</tr>
<tr>
<td>Changing interests of users</td>
<td>It is really hard to predict the changes. Also some people may read the news not because he/she interested in the topic in general but because she found it important.</td>
</tr>
<tr>
<td>Scalability</td>
<td>The news recommender system should have a fast and real time processing capabilities.</td>
</tr>
<tr>
<td>Context-awareness</td>
<td>The recommendation should be aware of context-aware changing user need.</td>
</tr>
</tbody>
</table>
6.2 MICAA-based context-aware news recommender: explorations

With regards to above stated challenge, MICAA solution brings together feedback-based recommendation with context-aware system. Both techniques challenge the improvement of the accuracy of news’ recommendation. In this sense, we introduce in what follows MICAA solution for context-aware news recommendation based on users feedbacks.

The proposal is aimed to show the support of MICAA’ theories and models on recommendation (Fig. 46).

The instantiation carries adaptation within a user-centered paradigm considering readers and their preferences as the main parameter for deciding and adapting recommendations (To what?). User social's feedback is acquired mainly from explicit feedback on different social networks (When?). Adaptations are processed and applied at runtime through a context-aware recommendation of news. The news relevance for recommendation is judged via a score-based identification involving users feedback (Where?).

Fig. 46 MICAA based recommendation: JouNum instantiation

The functional part of the system (How? What? Why?) depicts a similarity and scores-based adaptation. Defined social scores reveal the involvement of users feedbacks in a mixed-initiative recommendation. Accordingly contextualization capitalizes on an adaptation engine to ensure a
context-aware selection of relevant articles and a context-aware recommendation at runtime (Fig. 46).

The discussion of the solution is driven by the above-identified challenges.

- **Cold-start:** To our knowledge, only content-based recommendations do not need training data. MICAA is supposed in some way to follow collaborative filtering. Nevertheless, it alleviates this difficulty and support cold-start challenge through considering an editor's scores aimed to promote novel relevant items. The needs for editorial decisions do not mean that a recommendation will stay completely headed by journalist. These scores are limited to the immediate short term and they are supported by the item freshness in the first hours after its launching, when no social rank is provided.

- **Data sparsity:** In case of social news recommendation the risk of data sparsity is not critical, so the number of reader should not be a problem in itself. From news perspective the recency criteria contribute to the limitation of item numbers. On the other hand, the context-aware clustering of recommendation plays a part in resolving sparsity problem.

- **Recency:** it is a requirement for news recommendation. This requirement could be considered at different levels. First, the news-updating rate is of no particular relevance to the recommendation. The chance that old data may be detrimental to the predictive accuracy is paused due to the diversity of recommended data, their short live, and changing interests of users. On the other hand, our solution highlights feedback’s recency, and considers periodically computed scores in configurable time-intervals ($\Delta t$). We assume that considering only recent interaction for recommendation lead to adjust more rapidly and accurately to user changing interests.

- **User feedback/Changing interest of users:** feedback is a key point to involve users and consider their preferences at runtime. All MICAA applied solutions capitalizes on users feedback for modifying, adjusting and personalizing adaptation. Equally, feedback has a major impact on deciding recommendation and their validation. We believe that involving user for adjusting and assessing recommendation within an agile process may improve recommendation accuracy. Similarly agility maintains the benefits of considering feedbacks and improves the support of changing user interest at runtime.
• **Scalability:** The scalability of news recommendation necessitates well-designed algorithms to efficiently deal with large news corpus. Existing works proceed to clustering in order to optimize scalability (i.e. clustering users and limiting to the nearest-neighbor search or using the cluster centroids to derive the recommendations). These approaches, even though they can significantly speed up the recommendation engine, they tend to decrease the quality of the recommendations. We do not address explicitly the scalability issue; nevertheless we believe that context-aware classification of news and users contribute in some way the scalability. Since the consideration context allows to bound the list of recommendation.

<table>
<thead>
<tr>
<th>Recommendation Challenges</th>
<th>MICAA Support</th>
<th>JouNum Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cold-start</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data sparsity</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Recency</td>
<td></td>
<td></td>
</tr>
<tr>
<td>User feedback</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Changing interests of users</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scalability</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Context-awareness</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

• **Context-awareness:** Context-awareness is a challenging concept in different areas and is a key point for our solution. For recommendation as well it is imperative to accurately determine what content needs to be provided (recommended) to a reader and when. Therefore the recommendation rating function becomes:

$$ R : User \times Item \times Context \Rightarrow Rating $$

where User and Item are the domains of users and articles respectively, Rating is the domain of ratings, and Context specifies the contextual information where the interaction occurs. This requirement is at the origin of our solution. Observation on user behavior was consistent for decision-making in different field and proved the relevance of context data for making accurate predictions. For instance, on weekdays a user might prefer to read topicality, horoscope news when he/she logs on in the morning and the political/economic reports in the evening, and on weekends to read movie reviews and star interviews.
Table 19 summarizes MICAA support level of news recommendation challenges in order to better place the solution. We use Harvey Balls to show the level of cover for each requirement.

6.3 JouNum: news recommender

JouNum solution belongs to a research project on online news recommendation. Through this project, we take our idea as far as possible to meet the need for real time context-aware news recommendation. We made the assumption that the applicability of MICAA can be expanded to include a different UI concept: the content; specifically via a mixed initiative context-aware adaptation of news recommendation. The applicability of this solution is confirmed partially through the online journals of SUDPRESS. The end of the project expects a large-scale deployment and validation of the solution.

Our contributions are focused on proposing new avenues and strategies to be explored for news recommendation ensuring context-awareness and considering users feedbacks. The review of existing works and the elicitation of requirements oriented our thinking to social recommendation that capitalize mainly on users feedbacks to predict relevant news. Despite the variety of social recommender there is no common accepted definition on social recommender. A limited definition depicts it as a collaborative filtering based recommender considering online social relations as an additional input. Our solution is intended to put forward Mixed-initiative context-aware social recommendation. By combining these policies in practice, we assume that relevant recommendations based on their context, and feedbacks could be identified. The MICAA based recommender's proposal seeks to develop a set of scores and algorithms that result in a significant difference in recommending news with regard to the context of use and user's feedbacks.

As a first step, a preliminary study based largely on the analyses of online news, to learn about the behavior during news consultation and to identify requirements that could be exploited to carry out an improvement of news recommendations. Those observations serve to identify what interaction’s variance might incur during news’ item lifecycle. Four main behaviors were identified:

- **Normal** depicts the most common behavior, where the rate of consultation increases quickly and then normally decreases in the time.
- **Random** where the consultation rates do not depict a variation with a apparent logic.
- **Buzz** is about news that shows a huge variation at a given moment, not necessary during lunching.
Chapter 6. Context-aware Recommendation

- Monotonous this behavior is not redundant, and shows an almost stable consultation’ rates.

The project stakeholders, who are actively involved in the project agree that interaction on a news item is the main factor for assessing its relevance. We also expect that the role of user’s interactions (feedbacks) is not restricted to the measures of relevance. The users’ feedback goes far beyond simply denoting relevance. It also provides the opportunity to confirm the trust and to indicate clearly the degree of relevance or usefulness of feedbacks.

![Fig. 47 Potential behaviors of consultation.]

JouNum puts forward **Mixed-initiative context-aware** adaptation of recommendation increasing the accuracy of suggestions. Alongside with the contextual information’ that leads to better recommendations, users’ direct interventions are of quite importance for recommendation context-awareness [Hutchins et al. 86]. JouNum extends context-aware recommendation by incorporating user intervention during interaction session for personalization. We discussed above in the section 3.1 the relevance of considering users feedback and learning users preferences during interaction for adaptations.

Several techniques allow users to intervene and several advanced algorithms have been proven to recognize and resolve knowledge extraction issues and runtime adaptations. JouNum solution involves users in the context-aware recommendation loop and operates the following sequence:

1. Detect user context,
2. Compute Items rate with regard to context of user and users’ context,
3. Recommend news items considering context-aware scores based selection.
4. Analyze the user feedback on recommendation (promoting/demoting)
5. Update recommendation list.

In order to reach a proper evaluation, only explicit users’ feedbacks are considered to promote or demote an item for instance (selection of a recommended item, feedbacks on item displayed from recommendation). Then the context of recommendation (different background conditions in relation to the ongoing interaction of the user with the system) can be observed by the system such as time (from system clock) Location (from GPS) or Deduced from user's behavior (e.g., time spent on reading and/or adding an item to favorite indicates if user is working or not). A number of studies and projects are aimed at inferring, and predicting context (e.g., for mobile computing). Contextual information has different levels of importance, and not all of them are relevant for recommendation. Gathering relevant context information could be made in different ways; Manually, using domain knowledge of the recommender system’s designer, or automatically, by using feature selection procedures or statistical tests based on existing data. To determine news context we use an hybrid approach that define context based on a set of statically test applied on gathered data during interaction sessions.

Social context-aware recommenders take for granted that readers do not react to the same item in the same way for different context. For instance the context of reading influences the chosen item (i.e. at home user interest to an item can be more explicit via direct feedbacks (comment, discussion), which should not be the case at work. So JouNum assumes that the reputation of an item deduced from users’ feedbacks depends on their context. Accordingly the relevance of a news item depends not only on the item matching for users interest but also depend on another factor showing the suitability of the item for the users’ current context. This matching is called Item Context-aware Reputation (ICR), it compute the relevance of an item to a user in a given context. First the recommender identifies the context of user, and then the user context is compared to context of predicted items. First the contextual similarity of an item i to other items are calculated and the most reputed k items are selected as nearest neighbors. The similarity between two items i and j is computed using the cosine similarity as follows:

\[ \text{Contextual Similarity} (I,J) = \frac{\sum_c ICR_i \ast ICR_j}{\sqrt{\sum_c ICR_i \ast \sum_c ICR_j}} \]

Where ICPx is the reputation of item in a given context.
Determining the reputation ICR for an item follows a specific process. Fig. 48 shows main steps for determining context-aware reputation score. As a first step, a freshness score is defined in order to meet the recency requirement. «Freshness» aim to promote news that are published and/or edited only a short time ago. In order to validate the need for this score, observations have been made on 7000 articles. These observations give an idea on the life expectancy of an online news article. It consists on computing the average of duration where items are still visited by readers. Taking into account the nature of article life cycle, the freshness score is defined and computed continuously. The score is developed and supposed to determine, in a preliminary selection, items for recommendation. The freshness of a news article does not depend on the context, and it is mainly related to its creation, update and the publication time. Next code depicts the algorithm for computing freshness.

Once news’ freshness is confirmed, the corpus of articles is reduced and then a context-aware selection of recommended news become easier and faster. The recommender algorithm involves real-time context data to predict relevant new and then proceed to a context-aware selection of items. At this stage news items are rated with regard to their popularity for readers in a given context of use: Popularity × Context \( \rightarrow \) Rating.
Chapter 6. Context-aware Recommendation

---

Score : Freshness-Score(Item)

**Input:** creation date  
**Output:** Freshness  
\[ V = \frac{\text{System.Timestamp - Creation.Timestamp}}{60*60*24} \]  
**Item freshness** = \[ \frac{1}{V} \]  
*If ItemFreshness <= life expectancy Then*  
**ImportItem ()**  
*Else*  
**excludeItem ()**

The popularity score is computed continuously at runtime considering the number of displays of item in a given context matching the user context. Assuming that news popularity changes substantially within their life cycle, shorter periods (\( \Delta t \)) have to be taken into account in order to ensure that popularity is computed accurately. This also prevents the promotion of old news that have existed for longer duration and gathered several users feedbacks. The following code presents the method for calculating the context-aware popularity score.

Score : Popularity(Item)

**Input:**  
\( D: \text{nb display} \)  
\( C: \text{context} \)  
\( \Delta t: \text{period} \)  
**Output:** popularity  
\[ \text{Popularity} = \frac{D^\Delta_t}{D_t^\Delta_c} \]

The social rank consists on assessing the relevance of a news item by considering users feedbacks from different sources; internal in case when feedbacks are expressed in the journal web page itself and external on different social networks (such as Facebook, Google+, Twitter etc...). Social rate is an extensive score, so the implemented solution could capitalize only on internal feedbacks and/or gathering feedbacks from other selected external networks. Further the nature of feedbacks reflects different engagement level. We assume that the modality of feedbacks independently from its significance (positive/negative) provide in some ways an indicator on the relevance of item for the reader. For instance when a reader invests in commenting an item or joining a discussion about it, this implies that his/her interest rate is higher than people only reading or liking the same item. Taking into account this fact the ICR score defines different weight factors for each feedback. Once news
Chapter 6. Context-aware Recommendation

Items are rated and arranged with regard to the popularity and the context of use, a reputation ICR score is determined in order to reflect the rate of each item in a given context. This rate supports the determination of distance between displayed item and recommendations. ICR follows a classification of feedbacks with regards to their social rank. ICR score definition is both contextual and temporal in nature. In case of JouNum, gathering feedbacks include the journal web page, Facebook, Twitter and Google+ journal pages.

Score: Reputation-Score(Item)

\[1/(a + \beta + \gamma) \left( a \left( \frac{L_{ic}^\Delta + L_{tot}^\Delta}{L_{ic}^\Delta + L_{tot}^\Delta} \right) + \beta \left( \frac{S_{ic}^\Delta + S_{tot}^\Delta}{S_{ic}^\Delta + S_{tot}^\Delta} \right) + \gamma \left( \frac{C_{ic}^\Delta + C_{tot}^\Delta}{C_{ic}^\Delta + C_{tot}^\Delta} \right) \]

Apart from the basic objectives of recommendation research and its fundamental characteristics, its diversity of process and approach offers several avenues of exploration. The purpose of this chapter was to build on the strengths of MICAA framework for adaptation. We limited our investigation to news recommendation and we posed a detailed theoretical suggestion for a mixed initiative context-aware social recommendation with regard to the proposed MICAA frameworks. From a theoretical point of view it may be a good and even modernizing proposal. Since we believe that MICAA framework can support recommendation through different perspectives. However, attention needs to be given to the practical application, which is expected, by the end of the project.

6.4 Conclusion

This chapter presented methodological investigation of MICAA applicability for recommendation. This proposal is made in the context of the JouNum project and aimed to advance news recommendation for a Belgian local journal SudPresse. It is a subsidiary of the ROSSEL group and focus on the daily press in the French Community. The proposed algorithms provide new avenues for new recommendation that might explore with a view to addressing the challenges of context-aware social adaptation of recommendations.
Chapter 7 Conclusion

There is a growing realization that User Interfaces need to take the current context in account to be better adapted to the user’s needs. Building such context-aware UIs provides means for creating better user experiences. However, context-aware UI is not necessarily easier to use. More gadgets sometimes cause more complications and context-aware adaptation sometimes presents the most serious barrier for usability. What can we do to make sure that the increased potential of UIs helps users? Involving users, without overloading them, presents several benefits for a context-aware system [Lieberman et al. 00]. User interventions (feedbacks) form an important aspect of context. Further, the ability of a context-aware system to accept and react to those feedbacks will also increase user satisfaction with it. This exchange serves as a primary mechanism for controlling the learning behavior of the system [Lieberman et al. 00]. In this sense, this thesis aims to contribute the context-aware UI by a mixed-initiative adaptation approach reinforcing the support of user feedbacks for learning adaptation. We established an extensive conceptual framework for MICAA and confirmed our proposal through diverse concrete practical examples.

7.1 Context

The new emerging trend of innovative interfaces focuses on the way in which the interaction responds to user’s built-in abilities and the context of use. Contextual information of where and what the user task is, what the user knows and what the system is capable of can significantly make things easier. However this doesn’t mean that context-aware UI adaptation decisions are always right and meet users’ needs and expectations. Further, since only users can judge their UI, considering their preferences during interaction is a must. User intervention and feedbacks to a system come out an important aspect of context. They provide the interface a better understanding of the users’
Chapter 7. Conclusion

context, which is capable of minimizing mismatches between system’s decisions and the users’ expectations. There exists several works aimed to the same goal. But there is still plenty of room for improvement in this regard. This proposal is a step in this direction. This thesis addresses mixed-initiative context-aware adaptation as a mean of combining the positive aspects of the user feedbacks loop and learning-based UI context-awareness. The developed MICAA framework put forward an agile mixed-initiative context-aware adaptation.

7.2 Summary of the contributions

The contributions of this thesis, as stated in the chapter 2, are divided into three abstraction levels. We presented a mixed-initiative approach for context-aware adaptation (MICAA) of user interface. The approach is conceptualized and applied respectively to abstract UI identification, concrete IU specification, and recommendation. Table 20 shows the requirements detailed in chapter 2 related to the contribution of this research. Each requirement was addressed by one or more main contributions. Some contributions impact different requirements.

<table>
<thead>
<tr>
<th>Table 20. Requirement Coverage visualization using Harvey balls</th>
</tr>
</thead>
<tbody>
<tr>
<td>(poor <img src="image1.png" alt="poor" /> <img src="image2.png" alt="poor" /> <img src="image3.png" alt="poor" /> <img src="image4.png" alt="good" /> Good support)</td>
</tr>
<tr>
<td>C1</td>
</tr>
<tr>
<td>----</td>
</tr>
<tr>
<td>R1</td>
</tr>
<tr>
<td>R2</td>
</tr>
<tr>
<td>R3</td>
</tr>
<tr>
<td>R4</td>
</tr>
<tr>
<td>R5</td>
</tr>
<tr>
<td>R6</td>
</tr>
<tr>
<td>R7</td>
</tr>
</tbody>
</table>

We used Harvey balls to illustrate the empirical analysis about the coverage of requirements. First, the multi-dimensional conceptualization of MICAA (C1, C2, C3, C4 and C5) allowed to address main related requirements and contributes in addressing others ones. For instance, the integration of advanced logic for MICAA (R4), was mainly addressed by C3and C4, although implementations (C6, C7, C8 and C9) contribute in addressing R4 through providing different scenario of integration.

7.2.1 Conceptual contributions

The conceptual contribution consists on establishing an extensive computational framework. To cover MICAA of UI, we made an extensive
Chapter 7. Conclusion

review of mixed-initiative adaptation concepts and concerns. Several conceptual requirements besides technical ones are still challenging, which results in MICAA computational frameworks. First, given the importance of user interventions for MICAA, we developed an organizational taxonomy (UFont) in order to classify feedbacks. UFont is aimed to assess the importance of different types of user intervention. It served also as a useful first step to fill gaps in basic information on user intents. Then, we established a tripod framework intended for supporting the MICAA objectives. The framework consists on three conceptual features: meta-models, structure and a methodology.

- **MICAA-MM**: it consist on a conceptual meta-model that brings together all identified concepts for supporting MICAA. The metamodel establishes three main package covering different dimensions of adaptation (UIModel, ContextOfUse and AdaptationProcess). It is intended to benefit from any relevant expertise acquired in previous works.

- **MICAA-Str**: addresses the low governed relation between UI adaptation features. It presents the internally organized content for UI adaptation. The aim is to provide an extensive characterization for MICAA allowing the understanding of how adaptation concepts are considered and what is made up as a result. Adaptations are characterized regarding considerations (to what?), implications (where? and when?) and implementations (how? what? why?).

- **MICAA-MF**: establishes a procedural arrangement of MICAA, outlining the advantages of integrating agility with context-awareness. This integration is supposed to provides the adaptation process with improved user-centeredness, interactivity and responsiveness to user feedback.

Finally, to reach the goal of MICAA, we developed an evaluation framework to track levels of support, and outcomes for users. MICAA-IQS consists on a methodological structure for supporting the development of MICAA of UIs. The solution focused on two main points: the support of human behaviors for improving the reliability of interfaces and the consideration of dialogue principles ISO9241-110 to streamline the interaction.

### 7.2.2 Methodological and practical contributions

We address MICAA framework validity through diverse methodological and practical instantiations. First, from a methodological point of view, we established different algorithms in order to support MICAA convenience in three different application scopes.
Chapter 7. Conclusion

- First, for AUI identification, we outlined a MICAA based adaptation scenario and implemented it through a java application: M@RINA. It consists on a model-based environment for the generation of a graphical user interface. The main focus was to put into practice the principle of mixed-initiative context-awareness at the AUI level. A partial reification process was considered in order to allow an adaptation of potential next AUI (container) with regard to the current interaction and context. The adaptation decision capitalizes on a set of machine learning techniques and considers the user feedback.

- Then, we targeted the adaptation at a concrete UI level. We investigate MICAA for UI layout adaptation concerning the widget selection issue. A set of scores and algorithms was defined and implemented through a web-based prototype (WiSel). The implementation aims to support the layout adaptation through a context-aware widget selection. The web tool promotes user feedback for refining the adaptation decisions and allows the users a full control on the interface while assisting them by recommending appropriate interaction objects. Allowing them control on adaptation is intended to be in harmony with the whole interactive session without interrupting the main task in order to avoid overloading them.

- In a third phase, we initiated the reflection on further methodological instantiations for investigating MICAA for news recommendations. We addressed context-aware recommender systems by considering user feedbacks. A set of algorithms and scores was outlined aiming at improving news’s recommendation context-awareness and putting forward user feedback.

The goal was to improve the user experience with online journals by recommending relevant news with regards to the context of use and the user preferences. The algorithmic proposals are supposed to be subject of future implementation, tests and validation under the JouNum project.

7.2.3 MICAA’s Business Model Canvas

In this section we consider the Business Model Canvas to define MICAA business functions. The canvas is described as “a shared language for describing, visualizing, assessing, and changing business models.” It’s made up of nine building blocks that help focus attention on key attributes of a business. The canvas Map the business, vision and objectives, it identify various end users, features, development channels, integration points, client’s partners, competitors and Resources or skillset required to deliver.
MICAA consist on an adaptation approach for supporting context-aware interaction via runtime user involvement and machine learning. It offers a generic approach for supporting end-users during interaction, moreover MICAA can benefits different software and information system to provide their user more usable interfaces.

Fig. 49 The Business Canvas Model

To make a Business Model Canvas, we start by filling out the value proposition "what you do?". This helps keep the focus on the main goal as we fill out the other building blocks of the canvas. From there we build on that goal and determine how it can be achieved by adding details about the other activities and resources we have. MICAA business model can be represented over the business model canvas as follows (Fig. 50).

Fig. 50 MICAA Business Canvas Model
7.3 Limits and Directions for future work

In addition to the contributions that were advanced in this thesis, there is room for more work that could be the target of future efforts. In this section, we put forward some potential future work that can make user interfaces adaptation a less frustrating than they are today. The future is bright for integrating AI into UI adaptation decisions, combining the best of user-centeredness with the reasoning and recognition that AI has to offer.

Most of the related works and provided prototype consider a simple learning algorithms, although this fact can be considered as a limitation of the project, we believe that the contributions of this thesis are generic and flexible enough in order to accommodate also applications with more advanced intelligence (i.e. neural networks, genetic algorithms, and fuzzy logic).

Further despite the diversity of presented prototypes, their implementations do not allow the reuse of solutions. However the reuse of solutions can be an extremely efficient, saving time and resources during program development. We believe that an ERP system including application modules can enable continuous improvement across the system and help to maximize the approach productivity.

For our part, and in the current state of our research, the fulfillment of our ambitious objectives open the door to new opportunities and future tracks that can be explored.

- On conceptual level, it is interesting for the convenience to put together established practices as an authoring tool for MICAA application. Such environment is advantageous for advancing knowledge and facilitating the application of acquaintances. Further, common interface specifications are fundamental to allow infrastructures to link to each other easily.
- On the methodological level, integrating complex AI algorithms and UCD into interfaces: the future is bright for integrating more complex AI into user interfaces while keeping HCI’s “feedback loop” of observing users and designing systems around their need and preferences. Such integration will provide a great avenue for discovering the principles behind the smart context-aware user-centered adaptation that is hoped to achieve with user-interfaces. During this stage, key skills and challenges to apply AI-related know-how in practical adaptation situations are provided to stakeholder, nevertheless, only few techniques were applied for supporting MICAA. Further complex technique and adaptation scenarios have to be investigated.
Chapter 7. Conclusion

• A deeper experimental analysis of the implementation of projects can be planned, with the aim of drawing deeper and broader lessons for future application and providing a basis for policy development.

• Future investigations including broader applications domains are expected; specifically context-aware software process engineering [Ayed et al. 15] in order to cover the whole system’s development lifecycle. Further, cooperation with an ongoing work is foreseen aimed at instantiating MICAA for software development team.

Finally MICAA approach is not intended to only integrate adaptation approaches within a mixed approach. It is merely meant to help them in producing UI adaptation that fit the context-of-use better and meet user requirements, thereby involving end-users through their interaction without interrupting their main tasks. As a final future outlook, we believe that packaging MICAA models and modules for user interface adaptation as general-purpose products could increase their usefulness for real-life projects.
Author’s Declaration

The work presented in this thesis is an original contribution of the author. Parts of this thesis were published in the following peer-reviewed papers:

Conference Proceedings


Khaddam, I., Mezhoudi, N., & Vanderdonckt, J. Towards a linguistic modeling of graphical user interfaces: Eliciting modeling requirements. In Control, Engineering & Information Technology (CEIT), 2015 3rd International Conference on (pp. 1-7). IEEE.

Khaddam, I., Mezhoudi, N., & Vanderdonckt, J. Adapt-first: A MDE transformation approach for supporting user interface adaptation. In Web Applications and Networking, 2015 2nd World Symposium on (pp. 1-9). IEEE.


Author’s Declaration

Doctoral Consortium


Journals


Workshop Proceedings


Talks


Delivrables

Serenoa Project


JouNum Project


References

A

[Aas et al. 99]

[Adomavicius et al.11]

[Agrall 11]

[Akiki et al. 14]

[Akiki et al. 12]

[Akiki et al. 12]

[Akiki et al. 13]

[Alpaydin 04]

[Allen et al. 99]

[Allison 15]
References


[Amatriain et al. 09]

[Andersson et al. 13]

[Aoyama98]


[Arhippainen et al. 04]

[Assad et al. 07]

[Ayed et al. 15]

B

[Barber 10]

[Bardram 05]
References

[Basin et al. 10]

[Bevan 09]

[Bertini et al. 05]

[Beck et al. 01]

[Beaudouin 00]

[Beyer et al. 99]

[Blumendorf et al. 10]
References


[Bobadilla et al. 13]

[Bodart et al. 95]

[Bohen et al. 05]

[Bohen et al. 04]

[Boles et al. 01]

[Boles 10]

[Bouillon et al. 04]

[Breiner et al. 09]

[Breiner et al. 11]
Breiner, K., Bizik, K., Rauch, T., Seissler, M., Gerrit Philipp Diebold, P. Automatic Adaptation of User Workflows within Model-Based User Interface
References

Generation during Runtime on the Example of the SmartMote
14th International Conference, HCI International 2011, 9-14.

[Browne 86]

[Brusilovsky 95]

[Brusilovsky et al. 96]

[Brusilovsky 01]

[Brusilovsky et al 02]

[Buscher et al. 12]

[Bunt 07]

[Calvary et al. 02]

[Calvary et al. 03]

[Cao et al. 09]

[Chang et al. 09]

[Chapelle et al. 09]

[Chris 13]

[Chin 01]

[Chu 00]

[Chu et al. 04]

[Claypool et al. 01]

[Clerks et al. 04]

[Cockburn et al. 01]
References


[Continental 11]

[Coninx et al. 03]

[Cohen 90]

[Criado et al. 12]

[Damodaran,96]

[Dey 00]

[Dey et al. 01]

[Demeure et al.08]

[Dessart et al.11]
References

SIGCHI symposium on Engineering interactive computing systems. ACM, 2011. p. 95-104

[Dicheva 08]

[Dietrich et al. 93]

[Dingsøyr et al. 12]

[Dyba et al. 09]

[Eggemeier et al. 91]

[Eisenstein et al. 00]

[Ericsson et al. 80]

[Ericsson et al. 87]

[Evers et al. 12]
References


[Fréard et al. 07]
Fréard, D., Jamet, E., Le Bohec, O., Poulain, G. and Botherel, V. Subjective measurement of workload related to a multimodal interaction task: Nasa-tlx vs. Workload profile. In HCII'07, 2007, in press.

G
[1]

[2]

[3]

[4]

[5]

[6]

[7]
Genaro et al. 12

[8]

[9]
Glaiel et al. 13
References


[Gram et al. 96]

[Gruber 93]

[Guerrero et al. 09]

[Guo et al. 02]

[Gellersen et al. 02]

H

[Hart et al. 88]

[Hart et al. 88]

[Haykin 06]

[Herczeg 09]

[Hinekley et al. 00]
References


[Howe et al. 06]

[Hook et al. 99]

[Hook 00]

[Horvitz 99]

[Hutchins et al. 86]

[Hu et al. 08]

[Huan et al. 12]

[Huang et al. 01]

[Ilgan et al. 79]

[Irum et al.15]

[ISO 99]

[ISO 06]

[Ivory02]

[J]

[Jawaheer et al. 10]

[Jameson 05]

[Jameson 03]

[Jacob 06]

[Jameson 11]

[Jenning 93]
References


Kellen 03
References


[Kleyn 88]

[Knutov et al. 09]

[Knutov et al. 12]

[Kniewel et al.14]

[Korpi 12]

[Krogsæter et al.94]
Krogsæter, M., Oppermann, R., & Thomas, C. G. (1994). A user interface integrating adaptability and adaptivity, Adaptive user support: ergonomic design of manually and automatically adaptable software, L.

[Krulwich et al. 96]

[Krulwich et al.97]

[Kujala et al.05]
References

L
[Langlais et al. 05]

[Langley 97]

[Lane 90]
Lane, T. G. (1990). A design space and design rules for user interface software architecture (No. CMU/SEI-90-TR-22). CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST.

[Lavi et al. 10]

[Lewis 82]

[Lena et al. 16]

[Lee et al. 06]

[Leiva 11]

[Linden et al. 03]

[Lim et al. 09]

[Lie et al. 10]
References

[Liu et al. 03]

[Lieberman 94]

[Lieberman 00]

[Lieberman 01]

[Lieberman 09]
Lieberman, H. User Interface Goals, AI Opportunities. AI Magazine 30(4) 09.

[Li et al. 06]

[Luyten et al. 03]

[Maes 93]

[Maglio et al. 00]

[Maybury et al. 98]

[Mandyam et al. 02]

[Malliga et al. 11]


[Meshkati et al.11]

[Middleton et al.04]

[Mitrovic et al. 05]

[Mitrovic et al. 07]

[Mitrovic et al. 09]

[Motti 11]

[Motti et al. 12]

[Motti 2014]

[Mori et al. 04]

[Muller et al.97]
References


[Noor et al. 08]
References


O
[Oard et al.01]

[Obendorf et al.08]

[Oppermann et al.94]

[Oppermann et al.97]

[Oreizy et al.08]

[Owen et al.06]

[Ozgobek et al.14]

P
[Paterno et al.08]

[Paterno et al.09]
References


[Parasuraman et al. 00]

[Paramythis et al. 10]

[Patton 02]

[Peissner et al. 13]

[Pitkow et al. 99]

[Puerta et al. 97]

[Puerta et al. 99]

[Pu et al. 11]

R

[Rasmussen 86]
References


[Ramaprasad 83]

[Ralph et al. 80]

[Reid et al. 88]

[Reid et al. 89]

[Reid et al. 88]

[Rissland 84]

[Rizzo et al. 97]

[Rich 89]

[Rosman et al. 14]

[Rosson 84]
graphical user interface. Lexington, MA: Microsoft and Zenith Data Systems.

[Rothrock et al. 02]

[Rui et al. 97]

[Rubio et al. 04]

[Saxton 13]

[Scholtz et al. 04]

[Schutte 09]

[Scholtz et al. 04]

[Sears 93]

[Seffah et al. 05]

[Shneiderman 83]

[Shneiderman 97]


[Shneiderman 03]


[Silva et al. 12]

[Smith et al. 09]

[Spyridonis et al. 14]

[Stuerzlinger et al. 06]

[Szekely96]

T

[Terada et al. 04]

[Totterdell et al. 90]
References


[Tran et al. 12]

[Trafton et al. 07]

[Tripathi 08]
Tripathi, P Human-Centric Framework for Perceptually Adaptive Interfaces Proceedings of HCI 2008 The 22nd British HCI Group Annual Conference Liverpool John Moores University, UK 1 - 5 September 2008

[Trætteberg et al. 04]

[Triantaphyllou00]
Triantaphyllou, E. (2000). Multi-criteria decision making methods (pp. 5-21). Springer US.

[Tsang et al. 94]

[Tsang et al. 96]

[Tsandilas et al.04]

[Tsandilas et al.05]

[Tullis93]
References


[Turk et al. 02]

[U]
[Ullmer et al. 01]

[Usi 07]

[Uschold et al.96]

[V]
[Vastag et al. 94]

[Vanderdonckt 95]

[Vildich et al. 12]

[Vijayasarathy et al. 08]
References


[Whitla et al. 09]

[Wilson et al. 10]

[Wickens 77]

[Wickens 92]

[Wickens et al. 99]

[Wickens et al. 00]

[Wickens 08]

[Wolfman et al. 01]

[W3C03]

[W3C09]
References

[W3C10]

[W3C]
W3C consortium, Available at http://www.w3.org/

[Ye et al. 04]

[Zadeh 65]
Appendix

Appendix A. Model-based User Interfaces

Due to the wide range of application domains, aspects and contexts of use, it is not scalable for human programmers to create UI versions for each scenario, instead an automated solution is necessary [Gajos et al., 04]. In this sense, different models, languages, methods and software have been proposed facilitating the design, implementation and evaluation of interactive computing systems. Accordingly, UI development methods shift to Model-based Approach, which consists in iteratively generating the application, following different granularity levels for development, mainly four abstraction levels: Task and Domain, Abstract, Concrete and Final, allowing a tool to generate the specific version adapted for each device and modality, thus representing a viable alternative to overcome the limitations of other approaches.”[Mori et al. 04]. This Appendix provides an overview of the trends in model-based user interface’s design methods and illustrates theoretical frameworks, methods, languages and models that support model-based user-interface engineering.

1. Model-Based user interfaces design

There exist a lot of user interface approaches dealing with models dating back to the 1980's. Those model-based approaches deployed models throughout different UIs lifecycle, from the early UIs creation stages until their adaptation at runtime as well as design-time [Criado et al. 12, Paterno et al. 08]. Such approaches often use advanced modeling techniques like model-driven engineering (MDE) for the automatic generation of the UI. Commonly, to Address issues concerning the simplification of the process of user interface creation, and to provide an infrastructure to allow applications to run in different context was a major purpose [Gajos et al. 06]. To that end, different approaches have been proposed for adaptation problems; almost all of them stimulate adaptation via an adaptive behavior [Blumendorf et al. 10, Bodart et al. 95, Breiner et al. 09, Breiner et al. 11, Chu et al. 04, Mitrovic et al. 07, Criado et al. 12]. Generally for model-based approaches, the stepwise development life cycle put forward a separation of concerns providing a good basis for producing a well-structured system, besides facilitating implementation itself as well as maintenance. Adaptations have been applied over the UIs engineering process, in compliance with diverse abstraction levels defined by the Cameleon Reference framework (CRF) [Calvary et al. 03]. Investigations were focused on the definition of languages covering different abstraction levels describing the UI, rather the correspondences between levels (mappings) and transformation functions. Usixml is an example [Usixml07].
## Appends

### UIs models

<table>
<thead>
<tr>
<th>Model Class</th>
<th>Model Name</th>
<th>Model Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Archetypal</td>
<td>Task</td>
<td>Hierarchies of tasks that need to be performed in a specific temporal logical order for achieving users’ goals (during the interaction with the UI). It is a Platform Independent Model.</td>
</tr>
<tr>
<td></td>
<td>Abstract UI</td>
<td>Expresses the UI in terms of Abstract Interaction Units (AIU) (Vanderdonckt and Bodart, 1993), as well as the relationships among them. These AIUs are independent of any implementation technology or modality (e.g., graphical, vocal, gestural). They can be grouped logically to map logically connected tasks or domain objects.</td>
</tr>
<tr>
<td></td>
<td>Concrete UI</td>
<td>Expresses the UI in terms of Concrete Interaction Units (CIU) (Vanderdonckt and Bodart, 1993)). These CIUs are modality-dependent, but implementation technology independent, thus platform specific (PSM). The CIU concretely defines how the UI is perceived and can be manipulated by end users.</td>
</tr>
<tr>
<td>Ontological</td>
<td>Domain</td>
<td>Provides the special features for creating a user interface. These features are the attributes of the objects in domain model and the relationships between these objects.</td>
</tr>
<tr>
<td></td>
<td>Behavior</td>
<td>Describes how the dialogue is driven by the user’s (physical) interaction and includes the link between presentation and dialogue, such as clicking a button to pop up a viewer. Describe the interaction based on cognitive psychology and the work on formalizing human cognition and behavior. Which provide frameworks for evaluating how humans behave and perform when using a particular interface design.</td>
</tr>
<tr>
<td></td>
<td>Cognitive</td>
<td>Provide a classification hierarchy of user stereotypes, containing attributes and qualities that can be reference by e.g. mapping rules. Define the device in term of input and output capabilities, Attributes and qualities, which can be referenced by interaction object mapping rules.</td>
</tr>
<tr>
<td></td>
<td>User</td>
<td>Determine the set of physical, special, temporal aspects characterizing the context of interaction. Characterize the usability of a software product.</td>
</tr>
<tr>
<td></td>
<td>Platform</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Environment</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Quality</td>
<td></td>
</tr>
</tbody>
</table>
Moreover models can be exploited at runtime to recognize context changes and support on the fly adaptations [Blumendorf et al. 10][Breiner et al. 11]. Model based UIs are aimed to reproduce all potential benefits of model-based development and Model-Driven Engineering (MDE) in general. Almost all benefits and shortcomings of Model-Based User Interfaces were reported by W3C community in [W3C]. We would prefer the following advantages: (1) To reduce the gap between requirements and implementation: Since models are able to define precisely functional requirements to better match user expectations. (2) To improve communication between stockholders by explicit models, this explicitness enhances mainly for the use (understanding, perceiving, comparing…) of models and defines semantics of each model. (3) To enhance the productivity via the generation of code, the reuse and reducing errors,(4) To consider context evolution at runtime and enable on the fly adaptation.

HCI come across detailed and structured practices that make use of models in order to describe almost every aspect of the UI. The review of defined models within different existent approaches leads to belonging HCI models into two different classes archetypal and ontological [Calvary et al.03]. Archetypal models define the spine of a model-based approach, reviewed by the four abstraction levels of the camelion framework. The second class denotes different models that could enhance the UI description for instance context models, quality models, cognitive and behaviour models etc. Basically, three abstraction levels are identified by the spine of Cameleon frameworks, those levels are in line with the work of J. Vanderdonckt on OIA-OIC [Vanderdonckt 95] (Object Interaction Abstract and Concrete respectively).

A different classification was presented by Traetteberg [Traetteberg et al. 04], who classified models according to the semantic role of their design representations along three dimensions: the perspective (problem- or solution-oriented) of the representation, the granularity of the objects described and the degree of formality of the representation and its language.

7.3.1 Model-driven engineering of user interfaces

Today’s the number of software is in exponential evolutions counting advances in technologies and new potential scenario of use. Developing information systems, which is in line with the legitimate expectations is not effortless. Thus UIs advance is witnessing a “software crisis”, which was the name given in the late 70’s, to the problems experienced with developing software. The solution was to turn software development into a structured engineering discipline, with the purpose of developing qualified software supporting contextual changes, at a lower cost and with accurate results. Literature convoyed different approaches supporting the UI engineering. Most of presented approaches were proposed and addressed adaptivity and/or adaptability
Appendix

issue of user interfaces regarding the context of use. According to [Calvary et al. 03], these approaches are arranged into four categories: Interface Translation [Allen et al. 99]; Interfaces migration [Luyten et al. 03]; Markup languages-based approaches (USIXML) and model-based approach (Maria). Whereas the last categories has the advantage of applying the adaptation on models at height abstraction levels in a more flexible way. The term Model-Driven Engineering (MDE) is commonly used to describe software development approaches in which abstract models are created and systematically transformed into concrete implementations. As far as that goes for software, the engineering approach has been successful for constructing the user interfaces. In this sense, this section is limited to the presentation of model-based approaches on UI adaptation and UI usability. The Cameleon reference framework [Calvary et al. 03] represents a full integral framework of model-based UI adaptation as it defines four abstraction levels for the development of the user interfaces in a pervasive environment: tasks and concepts, abstract user interface, concrete user interface, and final user interface.

2. The Cameleon Reference Framework

As said above, the Cameleon Reference Framework [Calvary 03] was introduced to structure the methodology according to several levels of abstraction: the end users who are carrying out their interactive task on a certain domain of human activity, and a context in which the user is incorporated. From these, an abstract user interface is derived and then transformed into a CUI, generating a corresponding final user interface. The CRF makes explicit a set of UI models (e.g., Tasks, Abstract UI, Concrete UI, Final UI) and their relationships, to serve as a common vocabulary within the HCI Engineering community to discuss and express different perspectives on a UI. Relationships between the Camelion spine models involve for practices [Bouillon04]: concretization, abstraction, translation, and reflexion.

• Concretization: “reification” transforms a particular model into another one of a lower level of abstraction, for instance the transformation of the Task model is into an AUI model, which is concretized itself to a CUI. A CUI is then turned into a FUI, in general by means of code generation techniques stimulating a forward engineering.

• Abstraction: Conversely to the concretization, it transforms a UI representation from any level of abstraction to a higher one. Reverse engineering of user interfaces is a typical example of abstraction.

• Translation is a lateral operation that transforms a description intended for a particular context of use into a description aimed at a different context of use but at the same level of abstraction.
Appendix

- Reflexion is an operation that transforms a model into another one at the same level of abstraction for the same context of use (as opposed to different contexts of use as for translation).

3. User Interface Description Languages (UIDLs)

According to identified abstractions level for UI models, different languages were specified by different approaches in order to support the integration within development environments and to facilitate the work of designers and developers. For this purpose, the notion of User Interface Description Language (UIDL) has emerged in order to express any abovementioned model.

A UIDL [Gerrero et al. 09] is a formal grammar used in HCI in order to describe a specific UI independently of any implementation technology. The described UI include different interaction techniques (e.g., drag and drop, command line) and interaction modalities (e.g., graphical, vocal, tactile, multimodal). Commonly most of UIDLs are defined in a formal way as algebraic or model-theoretic structures. In MDE words, the UIDL is defined by a meta-model, which is a model of a modeling language. Such definition allows assuring the productivity of a language because it is in conformity with its meta-model. This assists the transformation of models.

A UIDL can serve at design time for the analysis of requirements as well as systems. It contributes to the refinement of the specification regarding the context. As well, UIDL can be deployed at runtime to realize a UI via a rendering engine. UIDL is a more general term than "User Interface Markup Language" (UIML).

4. The W3C meta-models for UI design

With the huge number of models, languages and approaches conveyed to support model-based UI engineering; the great challenge for the community was to be able to revive growth. In this sense W3C bring together all key stakeholders in the field who come to learn, to exchange and to share information, and that will allow them to anticipate the challenges of tomorrow for Model-Based UI (MBUI) context-awareness within an agreed upon a unified framework. The MBUI Working Group's initial focus is on models, UI components and integrity constraints at a level of abstraction independent of the choice of device. Future work is anticipated on engineering and context-awareness. Further out, The Commission is seeking to establish and introduce: a flexible, expansive framework for partners on the ground, enabling them to set up a set of standards for interoperable exchange of rules for dynamic adaptation to the context.
Appendix

Appendix B. Machine Learning Techniques

Investigation in the Machine Learning (ML) field resulted in the development of a wide range of algorithms. Typically, learning in these algorithms is accomplished by searching through a space of possible hypotheses to find an acceptable generalization of a concept. There are a wide variety of machine learning tasks and successful applications. For instance optical character recognition, in which printed characters are recognized automatically based on previous examples, is a classic example of machine learning. Although ML algorithms vary in their goals, learning strategies, the knowledge representation languages they employ and the type of training data they use. ML algorithms that do not require training are stated as unsupervised algorithms e.g. clustering and discovery algorithms that involve training with a set of pre-classified instances are referred to as supervised learning algorithms e.g. decision tree learning such as the Classification algorithm, neural networks etc. [Smith et al. 09].

1 Classical techniques

- Decision Tree. A Decision Tree (DT) is a predictive model allowing representing and classifying data. The goal of a DT is to create a model that predicts the value of a target variable based on several input variables. DT is a simple representation for classifying examples; it resembles the tree structure consisting of a set of nodes and a set of directed edges that connect the nodes. The main advantage of DT is that it is simple to understand and interpret. However such algorithms cannot guarantee to return the globally-optimal decision tree. A decision tree is employed in ContactFinder [Kruwihc et al. 95], InfoFinder [Kruwihc et al., 96), and Lifestyle Finder (Kruwihc, 97) to find and recommend contacts, documents and lifestyles.

- Decision matrix (DM). Basically DM aims to systematically identify, analyse, and rate the performance of relationships between sets of values and data. The DM is beneficial for large masses of decision factors and for assessing each factor's significance. DM is particularly powerful where we have a number of good alternatives to choose from, and many different factors to take into account. This makes it a great technique to use in almost any significant decision where there isn't a clear and obvious preferred option [Triantaphyllou 00].

- K-nearest neighbor algorithms (K-nn). It is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions). KNN served in statistical estimation and pattern recognition. It is an approach used for classification and regression. In both
Appendix

cases, the input consists of the k closest training examples in the feature space [Beyer et al. 99].

- Theory of fuzzy sets. It is a superset of established (Boolean) logic that has been expanded to handle the concept of partial fact values between "completely true" and "completely false". The logic underlines approximate modes of reasoning. The importance of fuzzy logic derives from the fact that it is inspired from the modes of human reasoning which are more approximate rather than exact in nature [Zadeh 65].

- Graph-based techniques. Refer to the set of machine learning models that adopts a fundamental graph structure. And then the graph-based models carry a more general sense than the graphical models, which appear in Bayesian analysis literature [Chapelle et al. 09]. Graph, among all representation forms, is a way to present data or knowledge. One major advantage enjoyed by graph-based models is the existence of structural information embedded in the graphs. Although graphs may not be able to completely uncover the possibly complicated structures, at least it provides a first approximation. There are a few potential research problems in graph-based learning models can be modeled by multiple correlated graphs, such as: The class graph and example graph in multi-label learning problem; The user graph and item graph in collaborative filtering; The user graph, query graph and document graph in retrieval [Li et al. 06].

2 Advanced techniques

- Neural networks. It is computational models inspired from central nervous systems (in particular the brain) that are in a position to machine learning and pattern recognition. They are usually presented as systems of interconnected "neurons" that can compute values from inputs by feeding information through the network. The model consists of a simple mathematical models defining a function f: X -> Y or a distribution over X or both X and Y, but sometimes models are also intimately associated with a particular learning algorithm or learning rule. The utility of artificial neural network models lies in the fact that they can be used to infer a function from observations. This is particularly useful in applications where the complexity of the data or task makes the design of such a function by hand impractical [Haykin 06].

- Reinforcement learning. It is more than a ML technique, it denote a paradigm of learning by interacting with an environment. It is concerned with the challenge how software agents ought to take actions in an environment and handle the trade-off between exploration and exploitation. so as to maximize some notion of cumulative reward. To obtain a lot of reward, a reinforcement-learning agent
Appendix

must prefer actions that it has tried in the past and found to be reactive in producing reward [Barto98].

• Relevance feedback. The idea behind relevance feedback is to enhance the user involvement in order to improve the final result set. In particular, the user gives feedback on the relevance of documents in an initial set of results. Relevance feedback techniques include query point movement [Jang et al. 13] borrowed from document retrieval literature, and weighting individual features and updating these weights heuristically in iterative retrievals [Rui et al.97]. More recently, optimization-based techniques that try to compute optimum weights [Rui et al. 97] or feature transformations [Huang et al. 01], and support vector machine-based techniques [Jing et al. 04], [Guo et al. 02] that use positive and negative feedback examples to learn to classify database images have become popular. However, optimization-based techniques are not applicable when the number of feedback examples is small and support vector machine-based techniques may also face stability problems due to small sample issues.
Appendix C. Mental Workload

With the growing complexity of User Interface (UI) it become crucial to incorporate more accurate measurement during the UI development in order to ensure a high usability and decrease the complexity of information systems. Different metrics related to the UI development and evaluations were advanced, however a notable lack of tasks complexity appraisal is still perceived. Weighting task is considerably valuable for modeling and adapting the UI rather than their evaluation. Instead of that, defined measures are defined for the evaluation phase to quantify accomplished tasks requirements. Rather most of metrics were subjective since the assessment of a task depends on different external factors such as user capability, and context of use. Assuming that weighing task in the evaluation phase was incredibly advantageous for analyzing systems in different field (e.g. military, laparoscopy…); and it has the same importance and usefulness to make use of it to guide the UI definition, we investigated here the workload measurement which offers an appropriated and valued measure for task weighting. Currently the evaluation of workload is a key point in HCI researches. It puts forward a decisive-summative tool for supervising the intention of adjusting systems and learning environments. Besides it were investigated for real-time feedback for adaptive systems. We focus on workload as a valuable criterion for weighting UI tasks and to guide UI adaptation.

Several works define workload as the physical and/or mental requirements associated with a task or combination of tasks. Wickens [Wickens 92] makes a significant contribution for this purpose, he states that the human operator does not have just one single information-processing source that can be tapped, but several different pools of resources that can be simultaneously recruited. Since that, multitasking has become prevalent in the society and cause several issues and dangers, multiple resource theories were defined according to this understanding and a 4-D multiple-resource model was outlined [Wickens 08]. The 4-D multiple-resource model is a design tool to predict workload. This model is considered as the base stone for many workload measurements. Therefore, it was upgraded according to new different new concepts as modalities [Boles 10], interruption [Trafton 07][Wickens 08] etc.

- **Workload Measures:** Mental workload measures are classified into three main classes [Meshkati et al. 11]: psychological, secondary-task measures and subjective.

- Physiological measures are based on both cognitive and autonomic activities and assume that mental workload is measured by means of the physiological-activation level. However, it is still complicated to select the evocative brain-signal for measuring workload.
Appendix

- Secondary-task measure or performance based measure reflect the earlier description of workload as inverse of spare mental capacity, which was excluded by several studies since interference of tasks is undesirable for their validity [Freard et al. 07].

- Subjective measures are, commonly, interview-based techniques aiming at reflecting the specialist impression of the mental effort required to perform a task. This workload measure category is the most applicable due to their practical advantages, for example: the ease of implementation, and non-intrusiveness subjective measures become the important and feasible tool for evaluation and task assessment [Rubio 04].

The literature review allowed us to identify the most outstanding existing techniques [Rubio et al. 05][Fréard et al. 07], which are the Workload profile [Tsang et al. 96], Swat [Reid et al. 88] and NASA-TLX [Hart et al.88]. The NASA-TLX tool provides a sensitive summary of workload variation [Hart et al. 88]. It applies a model of the psychological structure of subjective workload. The measure considers objective physical, mental, and temporal demands and their related factors. The procedure of measuring workload consists of two-step: first one rates the six previously described factors on a defined scale, the second one consists on evolving a comparison between scales aiming at weighting them appropriately to the measured task. Likewise, the Subjective Workload Assessment Technique SWAT was proposed by Reid [Reid et al. 88], it is based on three rating scale dimensions: Time load, mental effort load and Physiological stress load.

The psychometric proprieties of both techniques are well established based on many experiences and studies, such as [Rubio et al. 05], [Fréard et al.07] [Sandra et al 88] and they were exploited in a variety of fields as flight, aerial combat, remote vehicular control etc.

More recent researches identified a new measure to assess the Workload Profile (WP). The measure is based on the multiple resources model described above. It consists in a multidimensional instrument weighting subjective mental workload. Next table depict WP dimensions.

WP rating sheet extracted from Evaluation of subjective Mental Workload[Rubio04]

<table>
<thead>
<tr>
<th>Workload Dimensions</th>
<th>Stage of processing</th>
<th>Code of processing</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task</td>
<td>Perceptual/Central</td>
<td>Response</td>
<td>Spatial</td>
<td>Verbal</td>
</tr>
</tbody>
</table>

The tool assesses 0-1 rates for the amount of resources areas, mentioned in the table, during the tasks. The WP considers an ancestor version of multiple resources models cited previously.
Workload Measure selection: Within this variety of methods and metrics assessing the mental workload in terms of various workload indicators. Several studies proceed to evaluation in order to compare their capability. Accordingly, a set of criteria was defined to assess the effectiveness of proposed methods. For this reason, we reviewed selected criteria and we compared to restrain the most relevant one. Based on existing studies [Rebio et al. 04] [Freard et al. 07], we compare the performance of three tools mentioned above. The Table 4 cites most significant defined criteria according to [Rebio et al. 05], [jo 12], [Freard et al. 07];

<table>
<thead>
<tr>
<th>List of evaluation criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sensitivity</strong></td>
</tr>
<tr>
<td><strong>Diagnosticsity</strong></td>
</tr>
<tr>
<td><strong>Selectivity/Validity</strong></td>
</tr>
<tr>
<td><strong>Bandwidth</strong></td>
</tr>
<tr>
<td><strong>Intrusiveness</strong></td>
</tr>
<tr>
<td><strong>Implementation requirement/Reliability</strong></td>
</tr>
<tr>
<td><strong>Acceptability</strong></td>
</tr>
</tbody>
</table>

Several appraisals were established regarding criteria mentioned. For instance: Martin Schütte [Schutte et al. 09] compared thirty Methods for Measuring Mental Stress and Strain and classified the three previous measures as a psychological method that assures both reliability and validity. Fréad [Freard et al. 07] compared the NASA-TLX and WP according to analyses of dialog interaction conducted with a canonical discriminant analyses procedure. And concluded that WP is more informative than NASA-TLX.
Appendix

Comparison of NASA-TLX, WP and SWAT according to selected criteria

Moreover, [Rubio 04] compared the three measures by conducting two experiments: strenberg's memory and tracking task. She compared cited measures in pairwise in terms of their sensitivity, diagnosticity, validity, intrusiveness, and implementation requirements. Rebio's study was the most complete, since it considers several relevant workload evaluation-criteria.

Based on existing studies, we established a comparison between WP, SWAT and Nasa-TLX focusing on the above-defined criteria. We sum up obtained valuation from different studies. The previous Figure illustrates the obtained results.

WP was well behaved according to sensitivity, diagnosticity, and implementation requirements. On the other side, Nasa-TLX provided a skillful measure; as well it was marked by the greatest validity-criterion. Accordingly, NASA-TLX is promoted for use in the specification and conceptual level which need a height validity level to ensure estimations with respect to others. Thereby, NASA-TLX is the recommended tool for predicting the performance of an individual in accomplishing a task, while WP is more appropriated to meet the requirements of comparing task-mental-workload. Furthermore WP is more suitable for the analysis of cognitive demands comparably to the SWAT measure tool [Rebio et al. 05], [jo 12], [Freard et al. 07];
Appendix D. News recommendation

Recommender systems (RS) are attracting attention of several researchers and are beginning to take place in several fields. RS are deployed in a host of different applications throughout the field information retrieval. Recommenders are mainly aimed to help to cope with information overload by tailoring information to individual interests [Adomavicius et al. 2005], e.g., what product to buy (Amazon), what film to display, which publicity to promote etc.

Not long ago, the number of online journals has grown and the volume of digital information increases exponentially. In order to respond to this explosion, recommending online news articles raises new challenges. Several works addresses news recommendation via different techniques. Existing practices for recommendation do not meet requirement of news recommendation: for instance the lifecycle of items is short and identifying users is not possible. An extensive list of news recommendation challenges is presented by [Ozgobek et al. 14].

<table>
<thead>
<tr>
<th>Challenge</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cold-start</td>
<td>First-rater, ramp up, early rater: It denotes the problem when new items cannot be recommended and considered because they do not have any clicks from other users.</td>
</tr>
<tr>
<td>Data sparsity</td>
<td>The possibility of data sparsity increases if the number of columns or rows is much higher than the other (articles/readers).</td>
</tr>
<tr>
<td>Recency</td>
<td>It consists on the need for providing readers fresh news instead of old dated articles. So the relevance of news items decreases in time.</td>
</tr>
<tr>
<td>User feedback</td>
<td>The system should be able to collect user’s feedbacks effectively in order to identify their preferences.</td>
</tr>
<tr>
<td>Changing users' interests</td>
<td>It is really hard to predict the changes. Some readers can read the news not because he/she interested in the topic in general but because she found it important.</td>
</tr>
<tr>
<td>Scalability</td>
<td>The news recommender system should have a fast and real time processing capabilities.</td>
</tr>
<tr>
<td>Context-awareness</td>
<td>The recommendation should be sensitive for change of users context and consequently changes of user need.</td>
</tr>
<tr>
<td>Unstructured content</td>
<td>Some systems requires content information to compute recommendation, it is hard to analyze the content, especially for news. However, news items need to be structured and machine-readable.</td>
</tr>
<tr>
<td>User Modeling</td>
<td>The personalization of recommendations requires the construction of a user profile.</td>
</tr>
<tr>
<td>Gray sheep problem</td>
<td>Since collaborative filtering recommends items according to the user's common interests with other users, which does not mean the support of personalized recommendations.</td>
</tr>
</tbody>
</table>
Appendix

<table>
<thead>
<tr>
<th>Appendix</th>
<th>This is the problem when the system recommends similar or the same items with the already recommended.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Privacy Problems</td>
<td>Knowing about the users' preferences for personalization requires the storage of detailed data about the user and their analysis that can cause privacy issues.</td>
</tr>
<tr>
<td>Neighbor Transitivity</td>
<td>Neighbor transitivity occurs when the database is very sparse.</td>
</tr>
<tr>
<td>Synonymy</td>
<td>Separate resources can name same items differently and it is not possible for recommenders to understand that they refer the same item.</td>
</tr>
<tr>
<td>Diversity</td>
<td>Users tend to be more satisfied with diversified recommendations.</td>
</tr>
<tr>
<td>Trust</td>
<td>Trust can be built by a recommender system by explaining how it generates recommendations, and why it recommends an item.</td>
</tr>
</tbody>
</table>

The objective is to create a recommender system that is able to meet the real time constraints and to provide relevant suggestions. Further, the recommender system should be able to consider the context and be able to learn from user feedback.

As with the recommendation for other fields (friend recommendation on social networks, products for e-commerce etc) existing news recommendation approaches capitalize mainly on two techniques: the content based filtering and the collaborative filtering. Each technique has its own strengths and weaknesses. In the next table we review existing approaches for recommending news.

Recommendation Approaches

<table>
<thead>
<tr>
<th>Approach</th>
<th>Definition</th>
<th>Techniques</th>
<th>Challenges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content-based recommendation</td>
<td>CB recommends articles or commodities that are similar to items previously preferred by a specific user</td>
<td>cosine similarity measure</td>
<td>Gray sheep, Scalability,</td>
</tr>
<tr>
<td>Collaborative filtering-based recommendation</td>
<td>CF is the process of filtering for information or patterns using techniques involving collaboration among multiple agents, viewpoints, data sources</td>
<td>Pearson correlation-based similarity, constrained Pearson correlation-based similarity, cosine-based similarity, Jaccard metric</td>
<td>Gray sheep, Scalability, Data-sparsity Synonyms</td>
</tr>
<tr>
<td>Knowledge-based recommendation</td>
<td>KB offers items to users based on knowledge about the users, items and/or their relationships</td>
<td>Case-based reasoning, domain ontology</td>
<td></td>
</tr>
</tbody>
</table>
Appendix

<table>
<thead>
<tr>
<th>Hybrid recommendation</th>
<th>HR combines the best features of two or more recommendation techniques into one hybrid technique has been proposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic hybridization mechanisms (weighted, mixed, switching, feature combination, feature augmentation cascade and meta-level)</td>
<td></td>
</tr>
<tr>
<td>Cold-start, sparseness scalability</td>
<td></td>
</tr>
<tr>
<td>Social recommendation</td>
<td>SR offer opportunities for making recommendations by utilizing users' social ties, especially for systems whose rating data is too sparse to conduct collaborative filtering.</td>
</tr>
<tr>
<td>User-item rating matrix Breadth-First Search algorithm</td>
<td></td>
</tr>
<tr>
<td>Data-sparcity Trust</td>
<td></td>
</tr>
<tr>
<td>Computational intelligent recommendation</td>
<td>CI are widely used to construct recommendation models.</td>
</tr>
<tr>
<td>An artificial neural network (ANN), Clustering Genetic algorithms, Fuzzy set</td>
<td></td>
</tr>
<tr>
<td>Uncertainty gray-sheep problem</td>
<td></td>
</tr>
</tbody>
</table>

Discussion

While collaborative filtering based approaches considers user similarities, content-based filtering considers items similarities. Identified challenges present evident criteria to be taken into account when determining which recommendation technology to use for a given context. However the current recommendation system still needs improvement for present and future requirements of better recommendation qualities. News recommendation is an individual case and has specific constraint that should be considered.

We are being asked to devote greater attention to the news’ recommendation requirements and to listen to news-readers at first, because they usually speak very clearly. Further, we confirm that by considering the user’s needs, preferences and intents acquired through feedbacks, we can generate a more valuable recommendation list. To do this, not only do we need to understand user tasks, we need to rethink about how to involve user for conducting as well as evaluating recommenders.
Achevé d’imprimer sur les presses de l’imprimerie CIACO
Louvain-la-Neuve, Belgique
Juin 2016