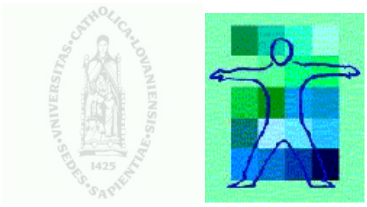


TriPlet: A Conceptual Framework for Multidimensional Adaptation of User Interfaces to the Context of Use

By Vivian Genaro Motti

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

Doctor of Computer Science
of the École Polytechnique de Louvain
Université catholique de Louvain

Committee in charge:

Prof. Jean Vanderdonckt, Université catholique de Louvain, Belgium, Advisor
Prof. Gaëlle Calvary, Grenoble Institute of Technology, France, Examiner
Prof. Víctor Manuel López Jaquero, University of Castilla-La Mancha, Spain, Examiner
Dr. Adrien Coyette, Université catholique de Louvain, Belgium, Reader
Dr. Hildeberto Mendonça, Université catholique de Louvain, Belgium, Reader

November 13th 2013

'C'est le temps qui tu as perdu pour ta rose qui fait ta rose si importante'
Anthony de Saint Exupéry

Acknowledgements

nanos gigantum humeris insidentes

Because none of this would be actual without the previous support of: my parents, my family, my colleagues, my friends, and all my teachers and professors, that by permitting me to *stand on the shoulders of giants*, gave me the opportunity to see the horizons from further, from different perspectives, from multidisciplinary backgrounds.

First and foremost, I thank my supervisor, Jean, for his support, guidance and inspiration during all steps of this thesis. I am heartily thankful for him accepting me as a research assistant and as one of his PhD students in UCL, and also for his trust in sending me for innumerous missions of the project; such experience certainly provided me growth and maturity for my whole career.

I am also thankful to my following committee, Adrien and Beto for accepting to supervise this thesis, for their time, support and for their valuable advice through this project, *un grand merci* and *muito obrigada*!

I thank the members of my jury, Gaëlle Calvary and Víctor López, who kindly accepted to participate of this PhD, as examiners. *Merci beaucoup* and *muchas gracias*!

I thank the master and trainee students who closely collaborated with this project, and with who I had the opportunity to guide and to supervise. In special, Damien Libouton, Mathieu Pas, François Debande, Quentin Poncelet, Mathieu Zen, Ousama Jammal, Raphaël Schramme, Denis Cariat, and Romain Van Vooren.

I thank my colleagues of LILab, Ugo for his friendship, support and suggestions, muito obrigada!, Diane, Nesrine, Suzanne, Sophie, Coralie, Mathieu, Iyad, Pascal, François, Jérémié, Sascha, Ilkka and Marco for all their support and collaboration: *merci à vous*!

Agradezco a la comunidad latina de Louvain-la-Neuve, en especial a Diana por su amistad sincera, *joie-de-vivre*, y chévere compañía ☺.

Agradeço a meus amigos bauruenses, de uma vida toda, Natália e Gustavo, Déborah e Felipe, Dani e Bruno.

Ik bedank mijn schoonfamilie hartelijk voor haar open armen, eerlijke glimlach, en voor alle zalige weekends die we samen doorbrachten. Er zijn niet genoeg woorden die beschrijven hoe dankbaar ik ben voor de gezellige momenten die jullie me boden tijdens de eindeloze regenachtige dagen en de lange koude winters in België.

Meu muito obrigada a minha família, meus pais, Adalberto e Telma, por serem guia, suporte e fortaleza, minhas irmãs, Lilian e Silvia, por sempre estarem perto mesmo estando longe, meus avós queridos, Encarnação e Antônio, e Mafalda, que me apoiam, me inspiram e que torcem por cada nova conquista.

Ik bedank Koen omdat hij altijd klaar staat om me te helpen, te luisteren, en te praten. Voor zijn eindeloos geduld, zijn hartelijke persoonlijkheid, en zijn eerlijke glimlach. Ook omdat hij me tot rust kan brengen in het midden van een storm, en omdat hij het beste in mij naar boven kan brengen.

This dissertation was realized thanks to the financial support of:

- Serenoa project that is funded by the European Union through its Seventh Framework Programme as a STREP Project no. FP7-ICT-258030
- ITEA2-Call 8 UsiXML (User Interface eXtensible Markup Language) project that is funded by Direction Générale DGO6 of Région Wallonne
- QualIHM (Région Wallonne, Direction générale operationnelle de l'Economie, de l'Emploi et de la Recherche DGO6)
- The WIST - Destine research project (Design & Evaluation Studio For Intent-Based Ergonomic Web Sites), funded by «WIST» Wallonie Information Science & Technology research program (Walloon Region, Convention n°315577, http://www.info.fundp.ac.be/DESTINE
- The GREENTIC JOUNUM research project, funded by Wallon Region

Abstract

The existing interactive systems tend to still consider a predefined context of use for interaction. Stakeholders mainly consider able-bodied users, desktop platforms, and stable environments. Conversely, users compose a heterogeneous group. They interact using different modalities and devices in distinct environments, which requires appropriate context-aware adaptations. Although adaptation has been largely investigated since the early 90's, its study has been often constrained. For instance by considering just one dimension of the context of use (i.e. user, platform or environment), by expressing and handling these dimensions in a restrictive approach by using only simple rules, by adapting just one aspect of an interactive system (i.e. content, presentation or navigation). Moreover, the end user benefit not always has the highest priority, making end users lost or without control over the adaptation. The existing frameworks about context-aware adaptation (CAA) are usually technologically driven, narrow in scope or obsolete. Due to these shortcomings, stakeholders have no unified support to rely on during the whole development lifecycle of context-aware adaptation. In order to address these main issues and aiming to bridge the gap between high-level adaptation goals and implementation of adaptation techniques, this thesis presents a conceptual framework for user interface adaptation that integrates the several dimensions that simultaneously compose the diversity of contexts of use through users, platforms, and environments, and the diversity of aspects of an interactive system, including its contents, presentation and navigation. This conceptual framework, named TriPlet, is structured in three core components: a meta-model (CAMM) covering an entire adaptation lifecycle, its concepts, properties, associations and constraints, a reference framework (CARF) that extensively defines adaptation concepts that support the design decisions for several application scenarios, and a design space (CADS) for consistently analyzing, comparing and evaluating coverage levels of CAA of user interfaces with well-defined criteria. A set of concrete applications demonstrates the usefulness of the adaptation concepts expressed by TriPlet, instantiating adaptation concepts as stated in the meta-model, in the reference framework and also in the design space.

Keywords: CAA; Context-aware Adaptation; Human-Computer Interaction; User Interfaces

Table of Contents

Table of Figures

Table of Tables

Acronyms

ACM	Association for Computing Machinery
ACAMD	Architectural Frameworks for Automated Content Adaptation to Mobile Devices
ADAPTS	Adaptive Diagnostics and Personalized Technical Support
AEHS	Adaptive Educational Hypermedia Systems
AJAX	Asynchronous JavaScript and XML
AUI	Abstract User Interface
AVI	Advanced Visual Interfaces
B&B	Bed and Breakfast
CAA	Context-Aware Adaptation
CAAUIG	Context-aware Adaptive User Interface Generation
CADS	Context-Aware Design Space
CaFT	Context-aware Frameworks and Toolkits
CAMELEON	Context Aware Modelling for Enabling and Leveraging Effective Interaction
CAMM	Context-Aware Meta-model
CARF	Context-Aware Reference Framework
CAWE	Context-Aware Workflow Execution Conceptual Framework
CFAWS	Conceptual Framework for Adaptive Web Sites
CHI	Computer-Human Interaction
CLG	Command Language Grammar
CF	Conceptual Framework
CTT	Concur Task Trees
COTS	Commercial-off-the-Shelf
COMETS	Context Mouldable Widgets
CRF	Cameleon Reference Framework
CSS	Cascading Style Sheets
CUI	Concrete User Interface
DIA	Digital Item Adaptation
DL	Digital Library
EICS	Engineering Interactive Computing Systems
EMUI	Existence of a Meta User Interface
EPG	Electronic Programming Guide
FAÇADE	Framework for Context-aware Content Adaptation and Delivery
FAHD	Framework for Adaptable Hypermedia Documents
FAME	Framework for Multimodal Adaptive Environments

FMIA	Fusion Framework for Multimodal Interactive Applications
FUI	Final User Interface
GPS	Geographic Positioning System
GUI	Graphical User Interface
HCI	Human-Computer Interaction
HTA	Hierarchical Task Analysis
HTML	Hypertext Markup Language
ICWE	International Conference on Web Engineering
idTV	interactive digital Television
IEEE	Institute of Electrical and Electronics Engineers
IQ	Intelligence Quotient
IS	Information Systems
ISATINE	Intention, Specification, Action, Transition, Interpretation, Evaluation
ISO	International Organization for Standardization
JCAF	Java Context Awareness Framework
LCAAF	Layered Context-aware Adaptation Framework
LF	Less Framework
MBUI	Model-based User Interfaces
MDE	Model-driven Engineering
MIF	Multimodal Interaction Framework
MOF	Meta-Object Facility
MVC	Model View Controller
OO	Object Oriented
OMG	Object Management Group
OWL	Ontology Web Language
PC	Personal Computer
PDA	Personal Digital Assistant
PHP	Hypertext Preprocessor
PUC	Personal Universal Controller
QoS	Quality of Service
RIA	Rich Internet Applications
ROAM	Resource Aware Application Migration
SDLC	Software Development Life Cycle
SQL	Structured Query Language
SLR	Systematic Literature Review
SMIL	Synchronized Multimedia Integration Language
SRG	System Recovery Granularity
SVG	Scalable Vector Graphics

TSC	Technological Space Coverage
TV	Television
UI	User Interface
UICG	User Interface Component Granularity
UID	User Interface Deployment
UK	United Kingdom
UML	Unified Modeling Language
UMUAI	Journal on Adaptive User Interfaces
USA	United States of America
UsiXML	User Interface eXtensible Markup Language
UX	User Experience
WG	Working Group
W3C	World Wide Web Consortium
XIF	XUL-based Interface Framework
XML	Extensible Markup Language
XSD	XML Schema Definition
ZIP	Zone Improvement Plan

Chapter 1 Introduction

The progressive growth in the amount and diversity of technological devices available on the market, besides enabling the interaction with many interactive systems, also increased the variety of user's profiles. As a consequence, today, the interaction takes place in several environments that are both physically and psychologically distinct. This growth in the availability of technologies summed with the easier access to Internet connections, easier mobility and portability of devices facilitated the interaction with many different interactive systems. Conversely, these systems are not always able to appropriately handle the specific requirements posed by such heterogeneity. Thus, there is still a significant gap between the actual needs of end users and what technology effectively offers them.

Aiming at reducing such a gap, the possible contexts of use from which the end users interact with needs to be carefully considered during the complete development lifecycle of interactive applications, so that the resulting User Interface (UI) of these applications is not only suitable for end users, but also convenient to work with in different environments using different devices and platforms.

Context-Aware Adaptation (CAA) aims at solving this issue by changing one or more aspects of an interactive system according to the context in which end users are located. The context involves any information that is relevant to characterize users, platforms, or environments. Such information is useful to define the necessary changes to be performed within an adaptation process. Multiple aspects of an interactive system can be subjected to these changes, including its navigational flow, its contents, and the UI presentation.

CAA, when efficiently implemented, provides users with interactive applications that are capable of adjusting their characteristics in order to ensure high usability levels.

1.1 Motivations

Most of the interactive applications often target at one specific platform and frequently consider that the interaction still occurs in a conventional context of use, i.e., an able-bodied user interacting via a unique platform in a stable environment. However, this *one-size-fits-all* approach is not suitable for the current technological scenario. Mainly because, users, besides composing a heterogeneous group, interact via different devices, from distinct environments and using different interaction modes (e.g. touch-based, pen-based, auditory modality).

Users rely on different platforms to consume media. They dynamically choose *when* and *how* they interact: a user can start to watch a movie on a TV at home, then switch to a smart phone on the way to work, and finish watching it on a tablet PC in the evening. Only by understanding the cross-platform user behavior and by providing a consistent experience for multiple platforms, users could have a more reliable interaction, effectively accomplishing their interaction goals. In 2012, nearly 33% of the page views in U.K. came from smart phones and tablet PC's [Com13]. Smart phones, PC's and tablets usage also varies according to time (Figure 1). The users tend to prefer to use their tablets between 8p.m. and 9p.m., their PC's during working hours (10a.m. to 5p.m.) and smart phones during commuting hours (e.g. early morning between 7a.m. and 10a.m.) [Com13]. Although

these statistics reflect the current reality specifically in the U.K., the multitude of devices is a global trend[1].

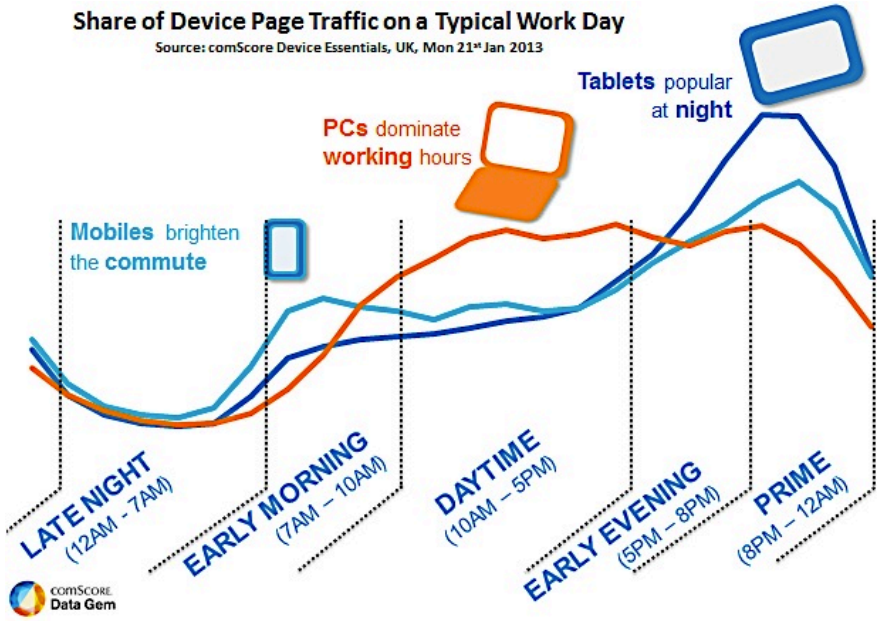


Figure 1. Multi-platform usage through the day: smart phones, PC's and tablets (U.K. February, 2013).

[Source: http://www.comscoredatamine.com/2013/02/an-average-monday-in-the-uk-pcs-for-lunch-tablets-for-dinner/]

Although 71% of the mobile users expect their applications to load as fast as on their desktop PC's, 57% of them have problems while interacting (concerning for instance performance and formatting issues) [Equ12]. Most of the mobile users already expect to make sacrifices in terms of content depth and feature-richness in exchange for the anytime, anyplace interaction. Besides this, they also expect anytime and quick interaction that works flawlessly. Mainly because of the urgent nature of their tasks, such as: checking flight status, comparing prices, finding locations of interest, confirming bookings, and making appointments.

The variety of scenarios in which the interaction takes place poses challenges for both end users and developers, and although researchers and practitioners have been investigating adaptation in a long term, there are still many open issues to be solved. Adaptation is a challenging and dynamic research field, motivating further research to progress in the domain and to aid to find effective solutions.

1.2 Contextualization

Given the current variety of contexts of use and the frequent absence of adaptation in various interactive systems, often the usability level of these systems is not high enough for end users, hindering or even preventing their interaction, and sometimes also requiring

[1] According to the data reported by the International Communications Market Report (at: http://stakeholders.ofcom.org.uk/binaries/research/cmr/cmr12/icmr/ICMR-2012.pdf)

users to adapt themselves to the application. An ideal system should adapt to the current user decreasing her mental and physical workload [Nor89].

From the *technological* perspective, there is a continuously growing offer of new devices, which on one hand, tends to provide better capabilities, but on the other hand also increases the fragmentation. Device's capabilities vary in terms of mobility (e.g. dimensions and battery life), network access (e.g. Bluetooth and Wi-Fi), quality (e.g. screen resolution and sound quality), storage capacity (e.g. memory cards and external hard drives), etc. For *developers* and *companies* it is difficult to quickly respond to the new requirements posed by this evolving landscape of new devices, mainly because there is no unified solution for this issue.

From the *end users* perspective, an interface should be flexible, plastic [Cal05] and also responsive [Mar10] enough to perfectly adjusts itself according to the context of use in which the users are located, i.e. optimizing the resource consumption, responding promptly, providing a high usability level, and also a great user experience.

From an *exploitation* perspective, mobile access has been continuously gaining momentum as a source of revenue. Therefore companies need to better understand how users actually access and interact in their different contexts. Delivering fast, reliable and responsive user experiences independently of the device or context of use is critical to embrace. However it benefits from the opportunity provided by increased mobile access [Com13].

Concerning the context of use, several information dimensions are involved. Users, platforms and environments have intrinsic interests and motivations to benefit from CAA, and also have different roles, as:

- Users expect an interface that perfectly suits to their profile, considering possible impairments, needs, wishes, preferences and constraints, and that also respects their confidentiality and privacy.
- The devices must have an optimal use of their capabilities, which requires their characteristics to be known and properly considered in the CAA.
- The environments provide valuable information for adapting the UI and enhancing the end user interaction. Sensors can retrieve relevant information, for instance concerning light, noise, and stability levels.

In spite of the major progress in CAA domain and its significant motivations concerning users, platforms and environments, still many open issues exist in this domain:

- The published frameworks that cover CAA are usually technologically driven, narrow in scope, or currently obsolete.
- The CAA studies are often constrained to one dimension of context at a time, i.e. only user, platforms, or environment is considered, and also to one aspect of an interactive system at a time, i.e. only the content, the presentation, or the navigation is considered. Once all these dimensions and aspects are important, they must always be covered in combination.
- The context dimensions are poorly expressed and handled, because their mutual influences are not always known, they cannot be properly prioritized.
- Simple rules are often adopted to process CAA. However, this expression format does not permit reasoning and inferences that are complex, involving priorities or probabilities, thus resulting in a reduced expressiveness and handling capability.

- Often other qualities of services are prioritized instead of the end user benefits. Given that CAA may cause disruption, making users to feel lost and without control during their interaction, the end users' benefit must be maximized and their disruption minimized.
- The users often do not have control to intervene in a CAA. Not only should the users be able to accept, reject, assess, or change adaptation rules, but also must the system be able to learn from their interventions adjusting its adaptation engine and improving its performance and accuracy.

Since the early 90's, works covering the different façades of adaptation have been reported. Progress has been achieved concerning: models, languages, softwares, and methodological approaches. Although there is an extensive list of works in this domain, usually they are not integrated, causing, among other issues, inconsistencies in the terminology adopted, difficulties in re-use, and incompatibilities in the solutions. Due to the lack of a unified methodology for determining *when* and *how* CAA must be implemented, the progress in this domain is considerably delayed.

Besides this, since the 90's new devices, technologies, and applications arose, evolved and became more popular, such as: idTV's (interactive and digital television), RIAs (Rich Internet Applications), and AJAX (Asynchronous JavaScript and XML). Therefore, not always the works that cover adaptation are sufficiently updated.

1.3 Definition

The word "adaptation" original from the Latin "*ad aptare*" (to fit to) stands for a change that makes something (e.g. a structure, a function or a behavior) suitable to a new situation. Adaptation is an adjustment necessary to suit new or special conditions, requirements, or needs.

For interactive systems, adaptation is a process that changes one or more of their aspects, for instance concerning the user interfaces (UI's), the adaptation occurs when certain contextual characteristics are identified and used to provide purposeful changes at the user interface [Bro86].

For Thies (1994) the increasing complexity of interaction demands for the users' assistance. Adaptive interfaces tailor the interaction of a system according to the individual user needs and to the environmental conditions [Thi94]. For Brusilovsky (1996), the adaptive systems reflect users' features in a user model and apply it to adapt various aspects of the system to the user [Bru96], for Lorenz *et al.* (2000) an adaptive system automatically adapts its behavior also according to the end user [Lor00]. Although, the user is the most important contextual dimension, it should not be considered solely. Also the platform and the environment play an important role in the definition of an adaptation process, hindering, or even preventing the interaction when ignored.

The context is the adaptation dimension [Bro86] used to define, guide and orient the CAA. In general, the context of use consists in any information that is relevant to define the behavior of an application [Dey00], including, mainly but not only, information about the user, the platform and the environment. According to Fischer (2012), the context awareness increases the resources on which systems rely to become more human-centered.

In an adaptation process, the changes affect certain system aspects. Such aspects are generally classified in three main categories: navigation, presentation and content [Bru01]. The navigation corresponds to the interaction flow, the hierarchical structure of

the user tasks, and their properties, as their order and requirements. The presentation varies in terms of modality type, and graphical aspects like contrast level, ergonomics and aesthetic metrics. The contents are represented by different UI components, such as: text, images, videos, audio and UI elements.

Within this thesis, we define the *context-aware adaptation* (CAA) as any change in the system aspects due to a specific context. Such a change includes adding, removing, modifying, or replacing one or more system aspects. The aspects here involve any component of an application, as contents, presentation, or navigation. And the context involves any characteristic of the scenario in which the user is located interacting with an application, as the device properties or the user impairments.

The main benefit of an efficient CAA is providing users more suitable applications, i.e., by sensing the context of use, and modifying the system accordingly, users should have a better usability[2] level and a better experience while interacting. More specifically, CAA aims at easing and simplifying the end user interaction by improving his or her performance, minimizing the need to request help, facilitating the system usage, simplifying the interaction with complex systems, and also avoiding cognitive overloads for the end users [Bro86], [Thi94], [Hoo97], [Lav10].

Although the main goal of CAA is to achieve better usability levels, there are many challenges and trade-offs involved in providing it. For instance: to perform adaptation further processing capabilities are necessary, first to constantly get context information and then to dynamically modify the application. This need of additional processing capabilities can negatively affect the performance of a given device. For Lum and Lau (2002), the critical issue in designing an adaptation system is how to determine a trade-off that guarantees the desired QoS (quality of service), while continuously getting context information, calculating, and properly executing adaptations. Besides this, when an application is automatically adapted, users can feel without control or lost if there is a significant rupture between the original and the adapted interface. Another challenge comes because context information is extensive, needing thus to be prioritized, however setting the right priorities and finding appropriate relevance levels for each context information is not a simple task. Prioritizing context is challenging because there is a large amount of context information that can potentially affect adaptation requirements, and the information that is the most relevant can vary according to each specific case [Moh06].

1.4 Shortcomings

There exists an extensive list of works that report about CAA, in both scientific and commercial domains. Some examples include adaptation concerning the type of platform (e.g., large screen devices [Neb11], mobile platforms [Des10], distributed interfaces [Gro05]), the interaction modality (e.g. vocal [Pat11]), application domains (e.g. entertainment [Pat99], e-learning systems [Bru99]), development methods (e.g. lifecycle [Bru07], [Lóp08]), qualities involved (e.g. plasticity [Cal05], the user experience [Arh09]), and type of content (e.g. video [Dra11]).

[2] Usability here is defined according to the ISO 9241, i.e. in terms of efficiency, effectiveness and user satisfaction [ISO9241]

Although adaptation has been extensively investigated since the early 90's, most of the studies are focused in one specific aspect of the adaptation at a time. For example: the architectural approaches [Han04], the platform types [Neb11], the user contexts [Kak10], or the content types [Dra11]. Although it is important to deeply investigate all adaptation aspects, without an approach that fully covers this domain, in an integrated manner, and using a unified methodology, many issues can raise.

From a *conceptual* perspective: (i) the terminology adopted by different authors is inconsistent or ambiguous and (ii) there is no standard abstraction precisely characterizing CAA. From a *methodological* perspective: (iii) the re-use of works is difficult, (iv) the extensibility of applications is complex, (v) the approaches currently adopted are not flexible enough and (iv) the results obtained are often not compatible. From an *empirical* perspective: (v) the solutions for CAA are scattered. And from a *programmatic* perspective: (vi) there is no agreed approach and a standard framework that can be universally adopted for implementing CAA.

Given the heterogeneous scenario in which the interaction with technology currently takes place, with multiple devices available on the market, a growing amount and variety of users, a facilitated access to technology, ubiquitous and pervasive applications, it becomes evident the need of applications that are prepared to efficiently adapt themselves according to context information. For Hanumansetty (2004) adaptation of user interfaces has become a necessity rather than a facility, and for Jankowska (2007), the necessity of CAA is inevitable.

Although adaptation has been largely investigated since the early 90's, there is no solution of broad coverage, tackling its multiple aspects simultaneously; due to this fact and also to the other shortcomings mentioned above, it becomes evident the need to investigate and to progress with CAA in a wider perspective, enabling the creation of methodologies that are capable of guiding, supporting and optimizing its implementation during its complete lifecycle.

1.5 Thesis

To address the aforementioned shortcomings, this thesis aims at:

proposing, defining, developing, and instantiating a multidimensional conceptual framework (named TriPlet). TriPlet includes a meta-model (CAMM), a reference framework (CARF), and a design space (CADS), that provides stakeholders with structured guidance for addressing context-aware adaptation of user interfaces.

A **conceptual framework** consists of a reusable design, composed, in this case, by a model, a reference framework and a design space and a set of implementations that define how to systematically apply such a framework (in a structured methodological approach). The framework covers not only the **multiple dimensions** of the **context information**, but also the multiple **aspects of interactive system**, i.e. any relevant information of the context of use, mainly concerning users, platforms, environments, and application domains, and also the aspects of a system that can be subject to the adaptation, as its contents, presentation, and navigation.

Context-aware Adaptation is composed of **different phases**, ranging from gathering the context information, to prioritizing, selecting, treating, processing, and using it in

an efficient manner. The involved strategies to support such phases aim at changing one or more aspects of the interactive system. By strategies, we consider: adaptation rules, principles, techniques, frameworks and models, i.e. all organized in a **structured methodological approach** that systematically supports and guides the definition and the execution of CAA.

Such approach includes the usage of a **meta-model** and a **reference framework** for defining and implementing adaptation, and a **design space** for analyzing and evaluating its coverage levels. The target stakeholders that can benefit from the framework support is mainly UI designers. However, such a framework can also be employed by other stakeholders, as developers, software engineers, and project managers.

The main goal of this thesis consists in identifying how to effectively support all development phases and design decisions of CAA, i.e. how to facilitate, with a structured guidance and support the specification, development, and analysis of CAA.

The goal of this thesis is achieved by: (i) performing a systematic review of the scientific literature related to CAA, (ii) identifying the main shortcomings in this domain, (iii) systematically analyzing and organizing its fundamental concepts, (iv) developing the support methods of the conceptual framework, namely a Meta-model, a Reference Framework and a Design Space for CAA, (v) proposing and implementing case studies to refine the methodological approach and to instantiate and validate the framework proposed, and finally (vi) analyzing, discussing and evaluating the outcomes produced.

1.6 Aims and Scope

CAA, by definition, involves different areas of computer science, such as: software engineering and architecture, distributed systems, human-computer interaction (HCI), and ubiquitous or pervasive computing. Although multiple areas contribute to CAA, this thesis will deliberately adopt the end user perspective first, therefore being mainly relevant to HCI. HCI concerns the design, the evaluation, and the implementation of interactive computing systems for humans use and studies a major phenomenon that surrounds them (ACM definition).

This thesis is inserted in a broad context of adaptation, i.e., the creation a conceptual framework (TriPlet) that is generic enough to be applied in multiple contexts of use, with platforms ranging from large screen devices, to desktop PC's, laptops, tablets, and smart phones and multiple application domains.

The main pillars of any development methodology for interactive systems include: models, methods, and software support. Interactive applications such as Information Systems (IS) do not escape from this structure.

Information Systems (IS's) concern an academic discipline that bridges the field of management sciences and the well-defined computer science field. IS as a new scientific area of study aims at investigating how computer-based systems support processes and structures within organizations.

For many years, IS's have been studied from a development lifecycle process that is very limited in scope and restrictive in expressiveness. We envision IS's as a field of study that is structured according to: one or several *models* that represent an information system at an abstraction level that is higher than at programming code; these models cover various aspects, such as: the end user, the computing platform, the organizational environment, the

tasks, the data, the processes. In order to support rigorousness, each model should be compliant with a *meta-model*, which regulates how models of the reality could be expressed. This meta-model is represented through a *language* thanks to which any model could be expressed, each language having its own *notation*. Our focus of IS study is the *end user interface*, with which users, groups, and organizations will interact in order to fulfill their business goals.

This methodology is built on three pillars: *models* (with their associated meta-models, language, and notation), a *methodological approach*, and a *framework* support to aid people to apply the methodology. We rely on the definition of the Cameleon Reference Framework (CRF) [Cal03] to structure the methodology according to four abstraction levels, and considering: the *end users* who are carrying out their interactive *task* on a certain *domain* of human activity, the computing *platforms* that they are using for this purpose, and the socio-organizational *environment* in which the user is located. From these, an abstract user interface is derived and then transformed into a concrete user interface, generating a corresponding final user interface.

For Nigay (1993), modality refers to the type of communication channel used to convey or acquire information. In the scope of this thesis the framework definitions are modality independent, however for the sake of simplification, for instantiating and validating the framework, the most frequent interaction modality has been chosen for analysis: the **Graphical User Interfaces** (GUI's) mainly because besides being the most commonly used, they are proved effective for different contexts of use and system aspects.

The framework proposed for supporting CAA developed in this thesis targets at any modality of user interface (UI): native applications, mobile applications and web pages. However the case studies are exemplified by means of **web applications** for the following reasons: the variety and access to the contextual information, about the user, the platform, and the environment at run-time to support CAA is simpler for mobile and web applications than for desktop applications. Even when CAA is defined at design-time, its application at run-time is more manageable (e.g. with a client - server - proxy architectural approach).

We work with the assumption that user interfaces for web applications can benefit from CAA by providing users an interaction with higher usability levels and a greater user experience.

Hypothesis. A conceptual framework facilitates the development of context-aware adaptation, by providing support and guidance through a structured methodological approach for stakeholders during: the design phases by means of a reference framework, the implementation phases by means of a meta-model and the analysis phases by means of a design space.

Scope. Although considering a large scope of interactive systems (domain, platform and technology independent) clearly represents a significant challenge, it is also noticeable the lack of (and the respective urgent need to have available) support methods (as methodological approaches and tools) that are capable of supporting the implementation of CAA as a whole, in a way that is sufficiently generic and flexible to cover the complete SDLC (Software Development Lifecycle), and to be applied in different contexts of use (context-independent), in different application domains (domain-independent), and regardless of technology (technology-independent).

Audience. TriPlet, the conceptual framework presented in this thesis has as its target audience user interface designers interested in context-aware adaptation. Both the scientific community and industrial practitioners can benefit from the framework.

1.7 Methodology

Figure 2 illustrates an overview of the methodology defined for this thesis. It is inspired in both the engineering method of Zelkowitz and Wallace [Zel98] and the iteration workflow of Koch [Koc01]. While the first proposes the development of a solution based on hypotheses and its sequential improvements according to tests results, the second proposes an iterative evaluation ranging from requirements capture, through analysis, design, and implementation to validation, verification and testing. In this project the validation of the results combines three of the approaches proposed by Zelkowitz and Wallace [Zel98]: literature search (as described in Chapter 2), case studies (as described in Chapter 4) and lessons learned (as described in Chapter 5).

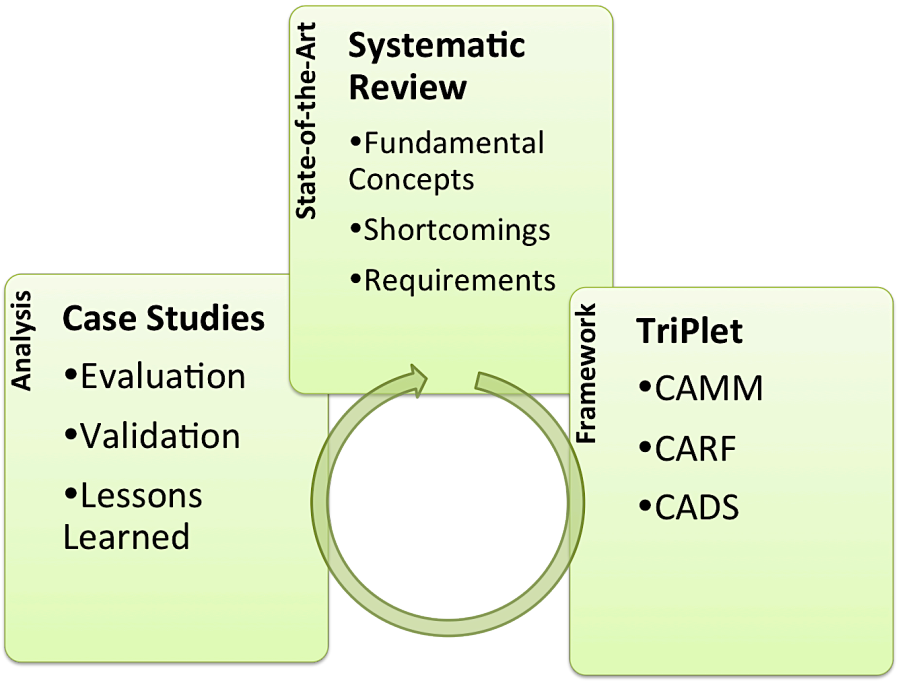


Figure 2. Methodology overview.

To generate TriPlet, the conceptual framework that supports the implementation of multidimensional CAA, three main phases have been accomplished. The first one consists of a Systematic Review of the literature (SLR) aiming at gathering relevant information. This information serves as a basis for the creation of the conceptual component of the framework (CAMM, CARF and CADS). This Systematic Review is followed by a second phase corresponding to the implementation of TriPlet. Then the third phase is executed. It consists in validating the approach and its feasibility with case studies that are also a proof of concept for the methods previously defined. Iterative evaluations have been conducted along this project to: critically analyze the results obtained, continuously detect potential flaws in the methods and results, and dynamically intervene to improve the outcomes.

1.8 Organization

This thesis is organized according to the diagram depicted by Figure 3, as follows:

Chapter 1 introduces the context of this thesis, motivates and defines its central goal, specifies working hypothesis and announces the thesis structure. It provides thus an overview about this thesis project;

Chapter 2 reports on a state-of-the-art of application domains, system aspects and contextual information that benefit from CAA of user interfaces, as well as models, frameworks and design spaces that supports its definition. It provides a fundamental background, important shortcomings and respective requirements;

Chapter 3 describes the methodology of this thesis, the conceptual framework created according to a set of shortcomings identified, its main components and how to apply them;

Chapter 4 reports the scenario of the case studies, their computational implementations and the respective instantiation and analysis of the application of the conceptual framework;

Chapter 5 explains the evaluation plan and discusses its resulting analysis;

Chapter 6 concludes this thesis by summarizing its contributions, discussing its shortcomings, and presenting future venues for it.

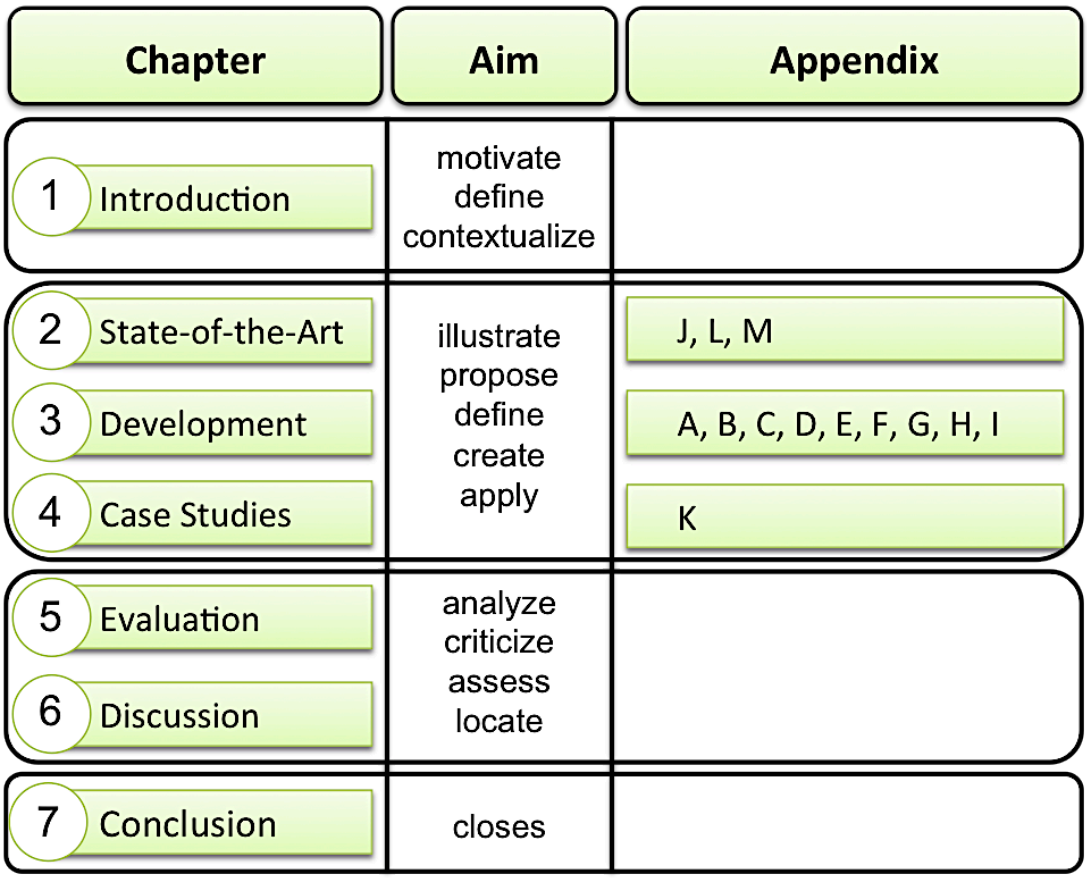


Figure 3. Reading map and thesis structure by chapter.

Chapter 2 State-of-the-Art

As presented in the previous chapter, this thesis aims at creating a conceptual framework that covers CAA regardless of: development phase, computational approach (architecture, technology), contextual information, system aspect and application domain. In order to fundament this thesis, we rely on existing works that have already been developed and published covering the domain of interest (CAA). This chapter presents the state-of-the-art in the domain of CAA, extensively summarizing and discussing related works.

2.1 Systematic Literature Review

The first phase of the methodology of this project consists in an extensive review of the literature regarding adaptation. By examining previously published studies [Zel98], we gathered the state-of-the-art of adaptation, gaining knowledge about: context information, dimensions, methods, techniques, strategies and principles that are relevant for CAA. A **Systematic Literature Review** (SLR) supports and formalizes the execution of this phase. In such review, an initial question is defined, and then its answer is searched with a systematic analysis of scientific documents that report related works. The goal is to identify the state-of-the-art in a specific topic, as well as possible trends and scientific gaps [Kit04], [Bre07]. Although a SLR is a time-consuming activity that requires significant efforts to be conducted, it is an appropriate technique in these circumstances, given that many documents need to be considered, analyzed and then consistently synthesized to extract and to organize the information of interest (adaption concepts). The main results of this SLR are presented in this chapter, they provided a conceptual basis for creating TriPlet, the conceptual framework proposed by this thesis and detailed in the next sections.

The review of related works published in the scientific literature is inspired in a systematic methodology as Kitchenham (2004) proposed. In summary, it consists in defining a research question ("*Which are the existing solutions for CAA, specially concerning: application domains, contextual information, system aspect, as well as, related models, frameworks and design spaces?*"), specifying the scientific bases for search (ACM DL, Journals as UMUAI, Publications in related conferences as CHI, EICS, AVI, or ICWE) and parameters for the research (search queries including: context-aware adaptation, meta-model, frameworks and design space), and defining a formal template to organize the information of interest (including keywords, contextual information and aspects involved, solution proposed, benefits, main contribution, reference information).

The related works were initially searched and selected based on their relevancy to define fundamental concepts, to identify main shortcomings, to motivate and derive the requirements for this thesis. The works reported in this chapter were also essential for creating the proposed framework: TriPlet. They are the current result of an extensive bibliographic review; still they represent a partial coverage of the related literature, once it is not possible to be exhaustive.

Due to the extensive list of publications in the domain of CAA, the works in this chapter were firstly selected according to their relevancy in this domain (based specially on the shared interests and similarity with the domain of this research) and secondly organized in six main sub-sections: (i) Domains, (ii) Aspects, (iii) Contexts, (iv) Meta-Models, (v)

Frameworks, and (vi) Design Spaces. Concerning an applied perspective: the sub-section *Domains* briefly illustrates four possible domains in which adaptation can be and has already been applied, the sub-section *Aspects* focuses on system aspects that are subjected to adaptation, and the sub-section *Context* presents existing works according to their contextual dimensions. Concerning a support perspective: while the sub-section *Models* describes existing models and meta-models for CAA, the sub-section *Frameworks* describes systematic approaches implemented to support decisions and development of CAA, and the sub-section *Design Spaces* describes existing design spaces related with CAA. Such organization is illustrated in Figure 4. Given that the analyzed works often cover multiple aspects simultaneously, just their main focus and contributions are considered to classify them.

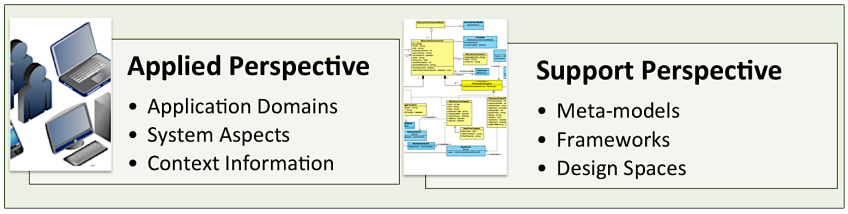


Figure 4. Organization of the state-of-the-art: applied and support perspective.

2.2 Applied CAA

To organize the state-of-the-art concerning the applications of CAA, first, we describe in this section possible application domains, and then, we provide examples of *what* in an application can be subject to it (system aspects), and finally we detail examples of usage for contextual information (according *to what* an interactive system can be adapted).

2.2.1 Application Domains

Interactive systems, independently of their domain, can always benefit from CAA, mainly because the user interaction per se can always be enhanced if the context is correctly considered. Due to the extensive work dedicated to CAA many domains have already been explored. For the sake of illustration, this section presents a brief selection of four domains in which CAA is widely employed.

For **e-Learning**, CAA has been applied to provide relevant and personalized contents to students according to their goals, interests and history (Figure 5 - left) [Smi02], [Spa03]. With such a student-centered approach, CAA tends to increase the efficiency in learning and the user satisfaction. For instance, when students have difficulties in understanding a specific subject, they access more detailed and illustrated contents.

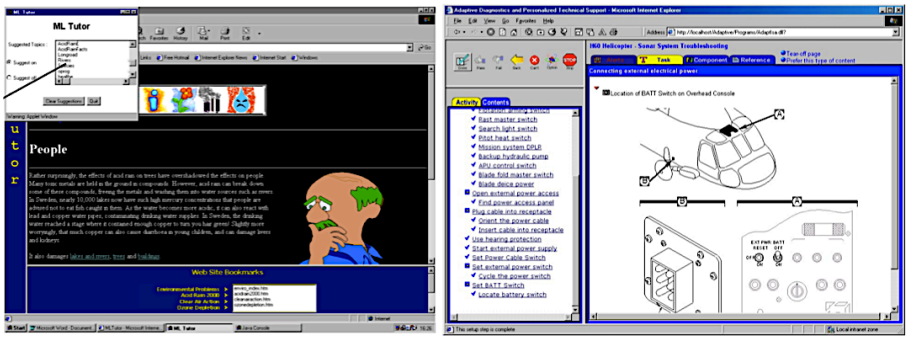


Figure 5. User Interfaces illustrating adaptation examples for an e-learning system (MLTutor's) [Smi02] and a safety critical domain [Bru02].

In the **e-health** domain [Rev00], CAA can aid users in medical situations by adapting contents according to the users' profiles. For instance, while providing guidance to a given treatment, an interactive system can adapt its technical vocabulary to the users' experience level (e.g. for a junior or a senior practitioner) and expertise domain (e.g. nutritionists or physiotherapists). As such, adaptive systems can provide many benefits for the clinicians' work [Bar03].

For **interactive and digital Televisions (idTV)** relevant contexts for CAA include: the profile of target users (e.g. age, preferences), their collaboration, their situation (e.g. relaxed vs. tense), their attention level, their frequency of use, the light level, the distance and the dimensions of the device, the characteristics of the remote control, and the availability of: a pointing device, an internet connection, a DVD reader or player, and a game console [Yha12]. CAA affects: the hierarchy, size, density and distribution of the UI contents (e.g. the users' choices). The electronic programming guide (EPG), for instance, can be easily adapted according to the users' profile, suggesting programs that match their actual interests.

For **Safety Critical Systems,** as air traffic control and nuclear power plants, an adaptive interface structure must be enough intuitive to clearly represent all necessary information. According to Acay, constrained environments require interfaces that are even more flexible and efficient. Complex application domains must have a small learning curve [Aca04]. Brusilovsky and Cooper (2002), for instance, investigated the potentials of applying adaptive and intelligent systems in the support of technical maintenance of aircrafts (Figure 5 - right) [Bru02].

Section 2.2.2 presents 3 main aspects of an interactive system that can be subject to CAA, regardless of application domain.

2.2.2 System Aspects

System aspects of different granularity levels can be affected by CAA, ranging from the complete application to a specific property of a UI element, as the font size. Figure 6 shows the taxonomy of Brusilovsky (2001) to classify adaptation techniques for presentation and navigation. Generally they involve presentation, navigation and contents [Bru01].

Navigation. Techniques that support adaptive navigation aim at helping users to find their paths in hyperspace by adapting *what* and *how* they can access contents according to their goals, knowledge, and individual characteristics [Bru94]. By adapting the navigation, an existing structure is modified or a new structure is created based on the contents and the context. Because it is a difficult task, it can only be performed automatically in some situations, e.g. when the navigation paths are pre-defined either by a map containing all links or by a table of contents with all headings. Typical navigation structures include: tables (lists), menus (hierarchies), and links (previous/next) [Jan07]. A navigation structure may be generated based on existing links or by creating links from relevant parts of the content (as: titles, headings, keywords). For instance, according to the age of the user, certain navigation paths can have their access disabled or enabled.

Presentation. Different devices and situations support different modalities and arrangements for the UI contents. Thus if the user requests a UI that uses a presentational mode that the device cannot render, or the situation does not support, the information will be inaccessible. Much UI content and multimedia types that applications use to present in-

formation, such as: maps to identify locations (e.g., the nearest restaurant) or description of products (for e-commerce) are unsuitable for certain devices. To tackle these problems, an adaptation system that changes the presentation (e.g. with a transcoding strategy) is needed [Lum02]. E.g. in a noisy environment the audio is inappropriate and a graphic modality is preferred, changing the approach in which the contents are presented.

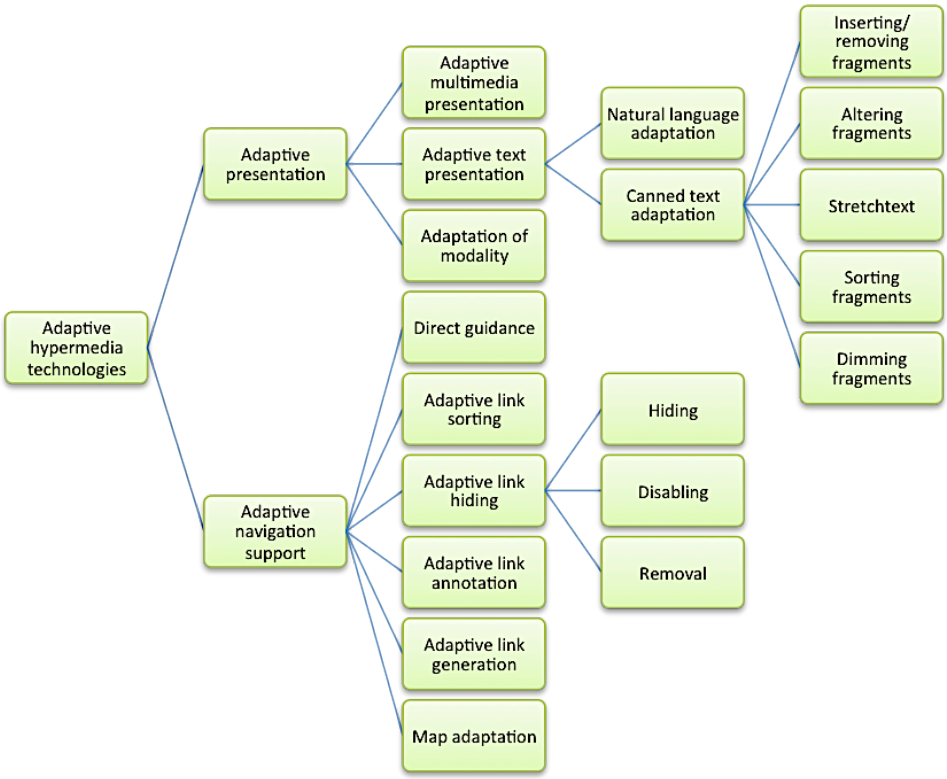


Figure 6. The taxonomy of Brusilovsky for adaptation of hypermedia technologies, including presentation and navigation aspects [Bru01].

Content. The UI contents have specific properties that are subjected to adaptation, be they semantic, syntactical, or physical. The contents can also be represented in different formats, such as: animation, graphics, images, text, videos, audio or UI elements, as forms, buttons, tables and icons. For instance, in a super wide screen, a text can have its content enhanced with complementary details and its font size increased.

The Appendix E lists 152 adaptation techniques targeted at modifying the navigation, the presentation, and the contents. Such aspects, targeted by CAA, are defined and briefly illustrated in this section.

2.2.3 CAA by Context Information

Although the classification of CAA techniques according to its three main system aspects is exhaustive, completely covering all techniques, it is hard to categorize each technique according to these classes and avoid ambiguity [Kap03]. Thus, in this Section the CAA examples are organized according to their target context information, covering different system aspects.

The fundamental unit that defines and orients the execution of CAA is context information. One must understand what context is, to determine its relevancy and how adaptation could exploit it [Abo99]. The context defines the selection of appropriate techniques and strategies for CAA, instantiating its several parameters. According to Schilit *et al.* (1994)

a context-aware application adapts itself according to the location of use, the collection of nearby people, hosts, accessible devices, and also according to the changes of such things over the time. Interactive systems with these capabilities examine the computing environment and react according to the situational changes.

Table 1. CAA examples and rules according to dimensions of the context.

Dimension	*Category*	*Properties*	*Classes*	*Scenario*
User	Profile	Age	Children Teenager Adult Elderly	If the users are elderly, the targets must be larger, and drag and pinch gestures must be preferred [Kob11]
		Attention Level	Distracted Regular Concentrated	If the user is distracted, then motion gestures may be preferred as the input modality [Neg12]
		Interaction History	Often access Regular access Rare access	If the menu items have a often access, they must be easily accessed [Gaj06]
	Disabilities	Cognitive	Dyslexia Autism	If the user is dyslexic, the font type must be adapted [Dir09]
		Visual	Blind Color-blind	If user is blind, then haptic feedback must guide the touch-based interactions [Gon11] [Oli11]
	Domain Expertize Level	Numeracy	Poor Medium High	If the user has poor numeracy, simplify the numerical information [Bau11]
Platform	Device	Network Bandwidth	Bits per second (bps)	If there are rate constraints in the network bandwidth, then remove channels or quality layers of the audio [Fei05]
		Type	Mobile Phone	If mobile phones, then the UI must support multiple interaction methods (focus-based, pointer-based and touch-based) [W3C10]
		Screen	Large Screen Wall Display Vertically Curved	If large screen, then optimize the use of screen space and reduce the scrolling [Neb11]
Environment	Environmental Factors	Temperature	Cold Warm Hot	If temperature is cold, then display advertisements appropriate for winter [Hea12]
		Noise Level	Loud Moderate Silent	If noise level is loud, then replace audio content with text [Fei05]
	Period	Date	Day Month Year	If commemorative date, then replace logo image [Doo12]

Relevant information about users, platforms, and environments can be acquired manually (explicitly provided by end users) or automatically (captured by sensors); via different sources, such as physical sensors (e.g., GPS, Bluetooth) or users' actions (e.g., browsing history, cookies, usage patterns) [Dey01], [Lor00]. Once the context is acquired, it must be firstly, assessed (e.g., regarding its confidence level, validity period, dependencies), and secondly, pre-processed (e.g., converted, treated or filtered). The context information can have different formats, such as: probabilistic, Boolean, discrete, or nominal, and it can be

modeled with different approaches, e.g., UML, OWL, Key-value, or markup languages [Str04].

Several works have been exclusively dedicated to investigate context information. This section summarizes CAA scenarios, illustrating different application domains, system aspects, and contexts. The Table 1 provides an overview of examples of 12 CAA rules, concerning context information belonging to: user, platform and environment and their given categories, properties, and classes.

The scenarios selected are merely illustrative, i.e. more complex scenarios may require the prioritization of further context dimensions to verify whether the rules are still valid or other ones are more appropriate.

More detailed categories are presented in the Appendixes (Appendix F, Appendix G, Appendix H, and Appendix I). The following sections detail the scenarios listed in Table 1.

2.2.3.a User

Adaptation concerning the users is commonly classified in two categories: adaptability and adaptivity. While the adaptability occurs when the adaptation is manual, i.e., when the user is responsible for it, the adaptivity occurs automatically, i.e., when the system is in charge of the process [Ste95]. Users represent the most extensive and complex dimension of context, mainly because they compose an information group that dynamically evolves and that includes innumerous characteristics, including for instance their:

♦ **Profile**. Frank and Szekely (1998) proposed and implemented a method to create adaptive forms, i.e., according to the content that the user provides to the application (age, gender, civil status), the original form fields can be expanded or collapsed. The approach is attractive for developers once they can produce domain-specific form-based UI's. AdapForms [Boh11] also enables the creation and validation of adaptive forms. It uses a specific *language* that designates structure and constraints to dynamically adapt the formularies presentation according to the contents provided by end users. Thus, depending on the users' input, the form can adapt itself, providing users more specific fields instead of an over-general form.

- **Age - Elderly.** Due to the disabilities imposed by advanced ages, elderly users face many challenges in interacting with applications. In a mobile context, such challenges are even more significant [Kob11]. Mainly due to the small devices' displays and buttons, and complex procedures. Despite of such constraints, touchscreen-based interfaces seem to offer senior-friendly interfaces, being thus generally easier for the elderly, and requiring few experiences to improve proficiency. Kobayashi *et al.* (2011) observed the elderly interaction with a touchscreen and created *guidelines* to support the development of senior-friendly interfaces. The guidelines define the use of: (i) larger targets (i.e., with 8 mm or greater), (ii) drag and pinch gestures (instead of tap ones), and (iii) a current mode status (e.g., for virtual keyboards).
- **Attention Level.** Distracted users, i.e. with reduced attention level, tend to prefer motion gestures as an input modality for smartphones [Neg12]. They compared the cognitive demands of motion gestures (tap, swipe or move) in two distraction scenarios (walking and eyes-free seated). Although the reaction times are not significantly different, the motion gestures require less time for checking the smart phone screen. Besides the gestures, the video speed can also be adapted according to the users' at-

tention level. Dragicevic *et al.* (2011) proposes an equation that varies the speed of playing videos in which, the beginning and the end of the video is exhibited slower than its middle part; thus calling the users' attention to perceive better the contents without overlooking them [Dra11].

- **Interaction History**. Ganneau *et al.* (2007) proposed an adaptive *menu* in which the items (i.e., contact names in a chat) change their orders according to the frequency of access; thus the most used contacts are firstly displayed. The same criterion was used to create an additional toolbar for a text editor [Gaj06] and can be used for adapting the navigation. To adapt the navigation, the most popular *techniques* are: direct guidance (i.e. when the system decides the most appropriate next step for the user interaction), sorting (i.e. ordering the links according to users' access), hiding (i.e. restricting the navigation space), and annotation (i.e. augmenting the existing links) [Bru94].

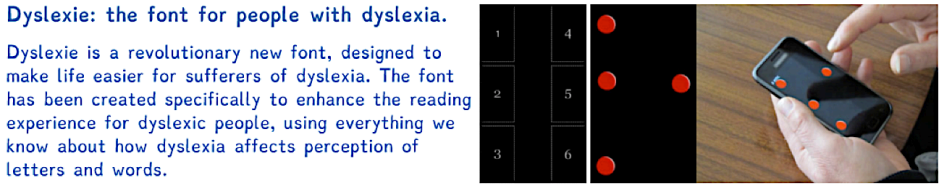


Figure 7. Interfaces for impaired users. Dyslexie is a font type that aids dyslexic users to read text (left). BrailleType maps screen regions according to the braille alphabet enabling text-entry for blind users (right).

♦ **Disabilities.** The design for all defends the universal access regardless of disabilities. Aiming to supports accessibility in all SDLC's, Kaklanis *et al.* (2010) presented a holistic framework able to simulate different user profiles and to perform automatic evaluation accordingly [Kak10]. The user models include physical, cognitive, behavioral, and psychological aspects. Such a framework aids in modeling the end user for creating adaptive interfaces.

- **Cognitive Impairment - Dyslexia.** Users with this disorder have problems in reading, spelling, and learning. The use of an appropriate font type[3] (Figure 7 - left) or of a different modality aids the text comprehension. Studies in an educational environment were performed and identified significant benefits of such adaptation [Dir09].
- **Visual Impairments.** Although a 'stereotypical' image of blind users is often considered, their individual differences impact their interaction and must be carefully considered. E.g.: their age, tactile sensitivity, spatial ability and short-term memory. In a mobile context, further constraints, as the limited input area, are imposed too. The mobile interaction requires a cognitive effort that, for someone lacking sight, is even more demanding. Although the spatial acuity impacts the time to explore the keypad, it seems to not affect much the interaction. The cognitive ability (measured with verbal IQ), on the other hand, has a significant effect on both users' effectiveness and efficiency. Mobile devices, specially concerning the keyboard characteristics, are challenging for all users, thus an inclusive design can promote benefits for all users. Once individual differences impact user performance, they must be considered in order to prevent exclusion [Gue11].

[3] http://www.studiostudio.nl/

- **Visual Impairments - Blindness.** GraVVITAS is a computer *tool* that provides an auditory and haptic interface of graphics for blind users. Graphics can be: a floor plan (map) and a line graph. The authors investigated the best reading *strategies* and *representations* for such information. The haptic feedback was provided according to the interaction in a touch screen [Gon11]. BrailleType proposes a text-entry method for blind users interacting with a touch screen mobile phone [Oli11]. The tactile feedback guides the interaction (Figure 7 - right).

♦ **Domain Expertise Level**. The adaptation according to the expertise level of the users, as proposed by Kantorowitz and Sudarsky (1989), considers that beginners and advanced users must have different interfaces and guidance levels to successfully interact with applications. UI elements, as menu items, and the interaction modality (e.g., audio) are the resources that can be adapted. The authors state that the freedom to adapt dialog models to actual users' needs is useful at all experience levels. Foss and Cristea (2010) applied the same concept in the e-learning domain. They proposed, developed and evaluated a *tool* that considers context information, as the expertise level and user preferences, to create adaptive courses based on existing resources.

- **Poor Numeracy.** Numerical information can be adapted, i.e., simplified, to improve its accessibility level according to the level of education or comprehension of end users. Guidelines help developers to replace numbers by equivalent contents that may be clearer for users with poor numeracy. Examples of *techniques* include: removing percentages and decimals and replacing the numbers by an equivalent textual description [Bau11].

The users' characteristics mentioned above are relevant for CAA. Besides those, for defining the CARF branches, other ones were also considered. They are described in the subsection 3.2.4 and illustrated in the Appendix H.

2.2.3.b Platform

The platform is characterized by the device (or set of devices) that the users interact with. Several characteristics of both hardware and software can be considered: screen dimensions and type, the battery level, processing capabilities, the network availability and configuration. In this section, we selected four characteristics that are enough representative to illustrate difference characteristics of a platform, and we briefly present CAA concerning: mobile devices, large screens, vertically curved displays and wall displays.

♦ **Device Characteristics.** Rousseau *et al.* (2000) defined design decisions that support the implementation of CAA for multimedia. Such decisions include using specific properties of SMIL for CAA for video. Audio applications for instance often need to handle several types of devices and networks, varying from low quality mobile services to very high quality home entertainment services. Ideally, a scalable audio format is adopted enabling to reach all formats for different devices. This requires a standardized solution that describes devices' and adaptations' capabilities, like content formats, network constraints, and user preferences, e.g.: the preferred volume, frequency equalizer settings, and audible frequency ranges [Fei05]. Varied scenarios can be considered: users with hearing impairments, learning a foreign language, noise level in the environment, device capability constraints. CAA techniques include: removing channels of the audio (e.g., from a multi-channel mode to a mono) or quality layers. The CAA of audio bit rates concerning device and network quality

of service, is handled by the DIA framework that optimizes operations according to a given set of constraints.

♦ **Mobile Devices.** The mobile landscape is heavily fragmented and to effectively support the development of mobile applications, W3C defined and published a set of 32 best practices recommended to facilitate the development and delivery of mobile web applications. To improve the user experience, they suggest for instance: allowing users to control the application behavior (as network and device data access), and compress content for more efficient delivery [W3C10].

- **Network Bandwidth.** Lum and Lau (2002) proposed a context-aware decision engine that decides the optimal content version for presentation and the optimal strategy for deriving and generating such a version. The engine takes into account the context (user preferences, client device characteristics, and network bandwidth) and the presentation (color depth, scaling factor and format). With a QoS-sensitive approach, it reduces a serious quality loss. The decision engine tries to reach the best trade-off for adaptation while minimizing degradation [Lum02].

- **Screen Types.** Several characteristics of the screens can be taken into account to adapt the UI. We provide below 3 illustrative examples for different screen types:

  - **Large Dimensions.** Most of the current UI's do not consider wide screen and high-resolution display (Figure 8 - left). By analyzing the spatial distribution of web pages UI's, Nebeling *et al.* (2011) identified scalability issues for large viewing sizes. Such issues underuse the screen space and demand unnecessary scrolling. Experiments with new technologies show that even the latest versions of HTML and CSS are insufficient for the effective CAA of content and presentation. The key challenges for this layout type are to reduce scrolling and to not overload the UI's with information. Although defining universal guidelines, ideal values and specific thresholds is a challenge, they created seven metrics to aid the analysis of the web layouts' quality concerning the view context, and to support the CAA of content and presentation for large screens. These metrics may also aid the evaluation of web layouts.

  - **Vertically Curved Displays.** The visualization paradigm of perspective+detail (Figure 8 - right) is an adaptation technique for vertically curved displays, defined by Schwarz *et al.* (2012). This technique extends the conventional overview+detail by adding a perspective viewing area and a text-based area with partial details. While the overview enables quick navigation through large information spaces, the detail view enables the access to relevant details. Figure 8 (right) illustrates this concept for rail traffic monitoring: the overview presents a railway network (each 3D dome represent a station and each color one fault type), and the detail shows textual information about the entire network and individual stations. A case study showed that: (i) the extended visualization of the overview supports users in different tasks, (ii) users understand the visualization and believe it is helpful, and (iii) users judge easier to accomplish their tasks (e.g. keeping their orientation in the network plan, or identifying the shortest route to the next target).

  - **Wall Displays.** The wall-sized displays provide a higher pixel density, simpler setup, and easier calibration. However, the resulting UI is often discontinued because of the frames of each monitor. To avoid the occlusion, distortion

and the non-legible parts of images, two interaction techniques were proposed, permitting users to access the contents behind the frames [Alm12]. While the CAA with ePan enables users to offset the entire image with gestures, with GridScape the head movements of the user are tracked to simulate motion parallax. Both techniques can be applied in multi-user scenarios.

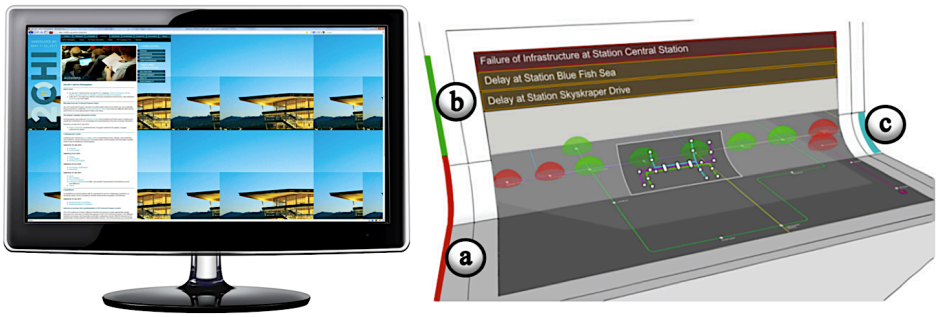


Figure 8. Lack of adaptation of the UI for a large screen (left) [Neb11]. Adaptation of the UI for a curved display (right). Perspective+detail: (a) Overview area on the horizontal segment; (b) Detail view in the vertical segment; (c) Head-up display on the curved segment [Sch12].

Besides the four platform aspects mentioned above, other ones can be equally considered. Further aspects are defined in CARF, as describes the subsection 3.2.4 and the Appendix G illustrates.

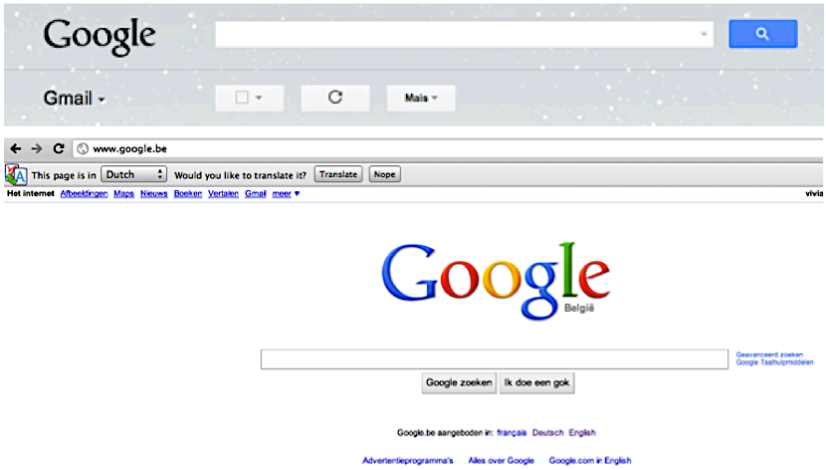


Figure 9. Gmail natural interface with the snow background adapted according to the weather (top). The Google search interface requesting adaptation of the language based on the location of the user (bottom).

2.2.3.c Environment

The environment is probably the most neglected dimension of context for CAA. Besides being complex, it is also expensive, once it requires additional resources and capabilities to continuously sense the context and also to dynamically perform adaptation. Even though, the environmental context provides rich information for adapting a system, as presented in the examples below:

♦ **Environmental Factors.** The patent 'Advertising based on environmental conditions', awarded by Google, defines that remote devices can get signal outputs from sensors, and such information about environmental conditions can be used to define advertisement contents [Hea12]. In other words, this patent concerns systems that allow advertisers to target

their on-line advertisements based on environmental factors of end users. The background of the UI for Gmail can be adapted according to the external weather (Figure 9 - top). Environmental factors include: temperature, humidity, light, sound, location of the user (Figure 9 - bottom), and air composition.

- **Noise Level.** Feiten *et al.* (2005) created a tool to adapt audio contents according to the context. Much context information is considered, including the level of noise in the environment. The noise frequency and level are used to dynamically adapt the audio signal provided. Besides this, an impulse response can be calculated, to directly adapt the rendering according to a complete description of the reverberation behavior in a room. The authors also envisage an extreme scenario, in which given the high noise level in the environment, the audio signal must be replaced by equivalent textual descriptions [Fei05].
- **Commemorative Dates.** Doodles is a Google project that adapts the logo of the search engine according to the day. Commemorative dates related to the user culture, such as anniversaries and seasons are marked with a dedicated version of the logo image [Doo12].

Other aspects that complement environmental dimensions of context are briefly described in the subsection 3.2.4 and illustrated in the Appendix F.

This section presents a series of adaptation that illustrate the potential of CAA. They are a resulting selection of the SLR, and as such they are covered by the conceptual framework, helping to define essential requirements and processes that support CAA in a broad perspective.

As previously mentioned, the context information is the fundamental unit to define a CAA process. Once it is known, it must be matched with a specific system aspect, and a respective adaptation technique, to perform the appropriate CAA. Normally, rules are employed to create such association. Such rules are defined in frameworks and models targeted at CAA as describes section 2.3.

2.3 Support for CAA

Due to the wide range of application domains, system aspects and contexts of use, it is not scalable for the human developers to create UI versions for each CAA scenario. Instead, an automated solution is necessary [Gaj04]. In this sense, different models, languages, methods and softwares have been continuously proposed to facilitate the design, implementation, execution and evaluation of CAA. This section presents a selection of 14 models, 21 frameworks and 12 design spaces that support CAA, and that inspired the design decisions and requirements for the conceptual framework proposed in this thesis (Triplet).

2.3.1 Models and Meta-Models

Models abstract system concepts and their relationships. In the domain of CAA, models have been mainly used to represent: context information, adaptation rules, and multimodal properties. This section briefly summarizes, in a chronological order, a selection of 14 models that cover specific concepts of adaptation. They are:

- **Munich.** is a reference model to define techniques for designing adaptive hypermedia applications. The domain model requires a conceptual design of the problem

domain, which evolves into a navigation and presentation model. The user model defines attributes and relationships with the domain model. The adaptation model specifies domain and user elements, the set of acquiring and adaptation rules and their collaborations [Koc01].

- **Customization Model.** Kappel *et al.* (2003) model the customization according to context regarding user profile, network and location [Kap02]. For them context provides relevant information about an interactive application and the environment. Context influences requirements elicitation and triggers customization according to context changes.

- **ADAPTS.** explicitly models task, domain and users, in an integrated manner aiming to support the adaptation according to the context. A diagnostic engine employs the user and expert models to update the navigation selecting the most appropriate tasks for the user based on pre-defined weights [Bru02].

- **Adaptation Model.** Vrieze *et al.* (2004) base on dynamic behaviors of the user to handling events and use ECA rules to pull and push adaptations in a more flexible fashion. It focuses on hypermedia systems [Vri04].

- **Context Information.** Fuchs *et al.* (2005) create a meta-model that defines context information and its associations. The main concepts considered include: devices and persons, their properties (as mobile phone, phone number, gender), and their relationships (as `is_located_nearby`, `has_phone_number`, `has_last_name` or `is_supervised_by`) [Fuc05].

- **Comets.** Calvary *et al.* (2005) state that an adaptation model specifies evolution and transition rules to be applied if the context changes. They propose adaptation models for defining tasks, abstract, concrete and final UI's and widgets extensions, always considering plasticity as the main principle. They remark the benefits of using model-based approaches to implement CAA, and they also emphasize the adoption of certain principles, namely: plasticity and continuity [Cal05].

- **CAWAR.** Fahrmair *et al.* (2005) propose a calibrateable context adaptation model for ubiquitous applications. It includes the context sensors, interpreters and also actuators [Fah05].

- **Java Context Awareness Framework (JCAF).** Bardram (2005) proposes a UML model for context including the abstract concepts of: an entity, the context, its items, and relations. It covers a person, a place, and a thing of an entity, as well as activities, status and locations of an item [Bar05].

- **User Model.** Kobsa (2007) identifies required characteristics for a user model for adaptation. They include domain independence, inference and reasoning capabilities, support for quick adaptation, extensibility, and privacy support. As future trends for this domain they remark ubiquitous and mobile computing, and smart appliances [Kob07].

- **Mobile Applications.** Farias *et al.* (2007) define a MOF-model for context-aware mobile applications. The concepts considered are abstract and include: classifier, attribute, entity, contents, associations, dependencies, groups and constraints [Far07].

- **Adaptation Rules.** Ganneau *et al.* (2007) and Sottet *et al.* (2007) create a meta-model that defines adaptation rules, targeting at plasticity as a goal for ubiquitous applications. This meta-model helps designers to take decisions and to implement CAA considering three phases: the context perception, the reaction, and the learning. The rules respect

the ECA structure (i.e., on event if condition do action). After the adaptation is defined, the users are able to request, accept or reject it.

♦ **Adaptation Rules.** López-Jaquero *et al.* (2009) and (2010) represent adaptation rules by means of a meta-model that includes concepts as: preconditions, events, sensors, data, transformation and transformation rules [Lóp09], [Lóp10].

♦ **UsiXML.** supports an MDE approach to cover all models required for user interface analysis and design, targeting the context of use, a dynamic entity, whose models are usually subject to continuous changes [Usi07], [Luy10]. Mainly the platform is taken into account, information considered include: the type of hardware (e.g. colors, sound output, text input, touch screen, keyboard), the network characteristics (e.g. capacity), browser type (e.g.name, version, html support), and the software type (e.g. handwriting recognition, and audio input encoder) [Lim04]. The meta-model of UsiXML [Usi07] for context information is shown in Figure 10.

♦ **Context of Use.** represents a generic model for context that was created for Morfeo project. This model integrates elements, properties, entities, aspects, components, characteristics, descriptions of environment and user [Mor12].

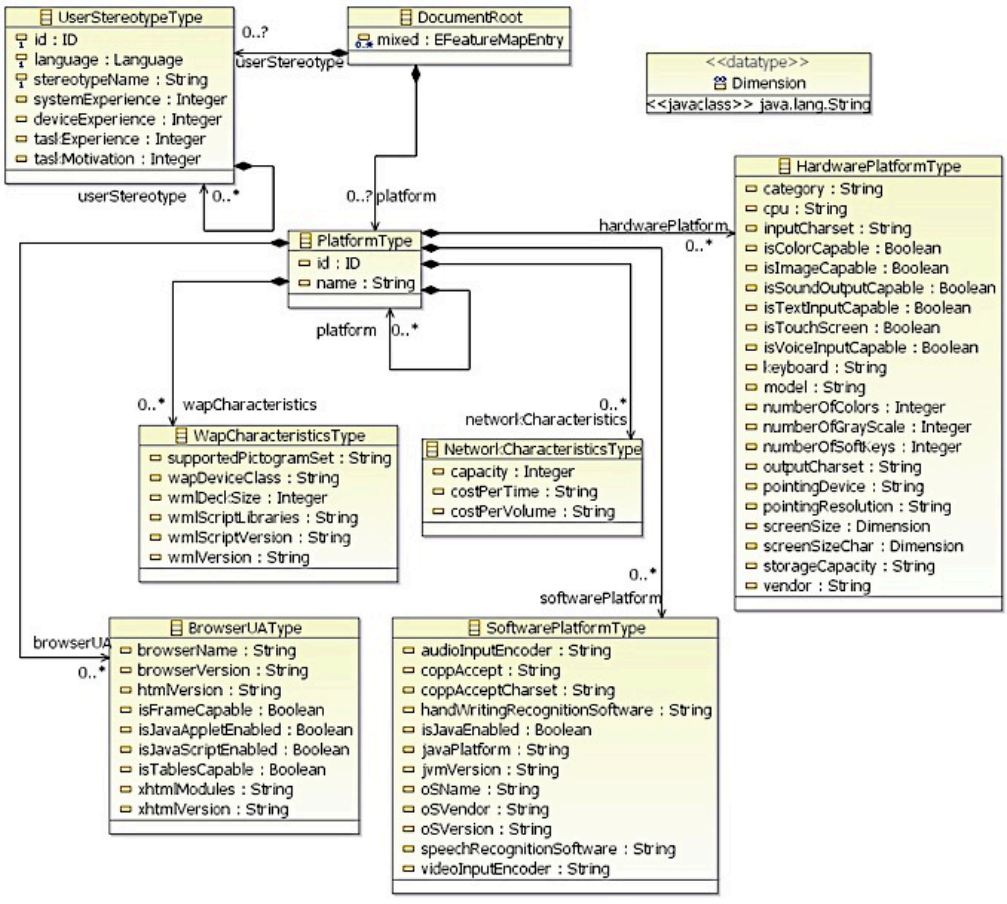


Figure 10. UsiXML model for context information [Luy10].

The models briefly presented in this section were selected based on their similarity with the topic of interest for this work, i.e. modeling CAA and also relevance for defining TriPlet. Once models analyzed target at specific concepts of CAA, they complement or specialize each other in a certain way. For instance, the meta-model of Farias *et al.* (2007) can be seen as a specialization of the work of Fuchs *et al.* (2005), Calvary *et al.* (2005) and [MOR12]. Although the works of Ganneau *et al.* (2007), Sottet (2007) and López-Jaquero *et*

al. (2009) and (2010) all focuses on CAA rules, the formers are more specific, respectively targeting at the adoption of principles and at the user feedback.

The meta-model diagrams proposed by [Koc01], [Cal05], [Fah05], [Gan07], [Far07], [Lóp10], and [Mor12] are presented in Appendix J.

Table 2 highlights the focus of the 14 works presented above. They can be broadly organized in two groups: while [Vri04], [Gan07], and [Lóp09] focus on rules, [Kap03], [Cal05], [Fuc05], [Fah05], [Bar05], and [Mor12] focus on context. More specifically [Koc01], [Bru02] and [Kob07] focus on user models, [Lim04] at platform models and [Far07] targets specifically at mobile devices. Such works are relevant to define essential concepts for adaptation; however by being specialized they provide a narrowed view of the adaptation lifecycle, i.e. by focusing in one specific part of the process, a global definition is still missing.

Table 2. Meta-models for CAA and their main focus.

Meta-Model	Focus
Munich Reference Model [Koc01]	User models (preferences, tasks, goals, experience) for adaptation, includes also rules.
Customization [Kap03]	User profile, network and locations.
ADAPTS [Bru02]	Task, domain and users. Tasks are then selected based on the user model.
Adaptation Model [Vri04]	Focuses on hypermedia systems, considers user models and ECA rules to pull and push adaptations.
Context Information [Fuc05]	Defines rules, context in terms of device and person, quality and associations.
CAWAR [Fah05]	Models the adaptation in terms of context interpreters, sensors and actuators, focusing on ubiquitous computing.
Comets [Cal05]	Context-aware adaptation and MBUI oriented to the plasticity of interactive systems.
JCAF [Bar05]	It covers the context, its items, relations and entities.
Generic User Models [Kob07]	User modeling systems characteristics of architectures, requirements and trends.
Mobile [Far07]	Context-aware mobile applications.
Rules [Sot07], [Gan07]	Evolution and transition rules based on context's changes. Adaptation rules for ubiquitous computing.
Rules [Lóp09], [Lóp10]	Rules in terms of conditions and transformations.
UsiXML [Lim04], [Luy10]	Considers the context as a dynamic entity and the platform characteristics mainly.
MORFEO [Mor12]	Context-awareness and the users' profile.

To identify which are the relevant concepts to compose the common ground for generating the CAMM (context-aware meta-model) the works reported here were analyzed

in depth and a summary of the results obtained with such an analysis are presented in Table 2 and Table 3.

Table 3 lists which are the concepts concerning specially: (i) Adapters (system, third-party, user), (ii) Context (user, platform, environment, application domain), (iii) Rules (justification, event, condition, action), and (iv) Models (task and domain, abstract, concrete, final) that have already been explicitly covered by previous works while modeling context-aware adaptation. The works on ADAPTS [Bru02], Generic Adaptivity Model [Vri04], and Generic User Models [Kob07] have not been included in Table 3 since they do not provide a meta-model of the process itself [Kob07] but task, domain and user models [Bru02], or an overview of the architectural structure [Vri04].

The check signs (✔) indicate when the concept has been explicitly presented as a class in the respective model, and the grey background indicates concepts that are expressed by a generic class, instead of representing all internal components.

Table 3. Concepts covered by meta-models for adaptation, based on their specificity (marked with the sign '✔') or generalization (marked with the grey background). The headers stand for: Adapter (System, Third-Party, User), Context (User, Platform, Environment, Application Domain), Rules (Justification, Event, Condition, Action), and Models (Task&Domain, Abstract, Concrete, Final).

	Adapter			Context				Rules				Models			
	S	TP	U	U	P	E	AD	J	E	C	A	T D	A	C	F
Koch, 2000			✔	✔						✔	✔	✔	✔		
Kappel, 2002				✔	✔	✔			✔		✔				
Fuchs, 2005				✔	✔	✔									
Fahrair, 2005											✔				
Calvary, 2005				✔	✔	✔						✔	✔	✔	✔
Bardram, 2005				✔		✔									
Sottet, 2007									✔	✔	✔				
Farias, 2007															
López, 2010				✔	✔	✔			✔	✔					
MORFEU, 2010				✔	✔	✔									
UsiXML, 2010				✔	✔										

By analyzing Table 3 we notice that most of the works targeted at modeling context-aware adaptation covers mainly context and rules, however the adapters, i.e. who is responsible for the adaptation is often omitted and the resulting models of the adaptation are also often not covered.

2.3.2 Frameworks

A framework is formally defined as a reusable, semi-complete structure that can be specialized to produce custom applications. It consists of a set of components extensible

for specific application domains. Frameworks are a proposition, which, if properly designed, reduce the investment and development costs [Jan07]. According to Bardram (2005) the frameworks' goal (mainly for context-aware computing) is to facilitate the development and deployment of context-aware applications. Stakeholders can focus on activities that are more specific for their application, while relying on a basic infrastructure to handle the actual management and distribution of this information [Bar05].

A framework helps to create explicit structures, which can be made complete and comprehensive by repeated investigations over time. It also contributes with a consistent terminology to facilitate sharing, to describe results, and to establish design guidelines and techniques [Sch04].

Once a framework can have different shapes, the works reported in the literature and analyzed in this project come in different formats, including: application toolkits [Dey00], [Gaj06], architectural approaches [Nic02], [W3C03], [Han04], [Jan07], [Pre09], [Mal10] conceptual definitions [Fis12], logics [Ard07], [But07], etc. Thus often they complement or specialize each other. And as such, it is a challenge to compare them. The frameworks presented and discussed below are dedicated to support CAA or closely related concepts. They are reported by chronological order.

- **A Framework for Adaptable Hypermedia Documents (FAHD).** [Llo97] works on interchange formats and languages to provide adaptation techniques (for layout, style, links, and synchronization). The context considered includes platforms, user characteristics and preferences. A generic model and a standard aid the transformations to multiple formats and target presentations based on one source document. This framework supports automatic processing of hypermedia documents to form presentations, building upon text-based standards, to transform structured documents. Specifications of presentation can be recorded in style sheets, and are broad enough to cover many hypermedia document sets.

- **A Conceptual Framework for Adaptive Web Sites (CFAWS)**. [Per00] focuses on the user model, navigation, and user views to adapt pages. While the user model aids to customize pages, the navigation is adapted based on the visit sequences. This work aims at improving the end user navigation by making it more efficient.

- **Context-aware Frameworks and Toolkits (CaFT).** Dey and Abowd (2000) define context information as relevant concepts that characterize the situation of the user. They defined requirements for implementing context-aware systems, and created a framework that eases their implementations by separating acquisition and delivery [Dey01]. It includes: widgets (that treat the context), interpreters (that extract meaningful information out of raw data), and context servers (proxy for communication).

- **Personal Universal Controller (PUC).** is an architectural framework composed of four parts: appliance adaptors, communication protocol, a specification language and interface generators. It controls real appliances and uses decision trees to render user interfaces in different modalities. It is a personal universal controller that enables users to control any appliance within their environment. The UI is automatically generated [Nic02]. As an input, the description of the appliance's functions is used. With user studies they noticed that end users preferred the automatically generated UI (instead of the manufacturer's UI's of the actual appliances).

♦ **W3C Multimodal Interaction Framework (W3C MIF).** is a generic architecture that identifies major components and respective functions for multimodal systems. The modalities considered include: speech, handwriting, keyboard and mouse. As languages, it is considered: XHTML, SVG, SMIL and HTML. Input and output modes are also taken into account. While input mode considers its recognition, interpretation and integration, the output mode considers its rendering, styling and generation. Use cases illustrate possible instantiations of this framework [W3C], [W3C03].

♦ **A Framework for Adaptive Educational Hypermedia Systems (AEHS).** Oliveira and Fernandes (2003) define a framework with 8 components. It gathers the context (information on learners' behavior), creates a domain model (learning theory), its sub-models (with main topics and sub-topics), a learning model (to define the instructional design theory), a hyperbase sub-model (with a meta-data library of learning objects, as exercises and presentation), the learner model (with the learners' characteristics and how to adapt to them), a decision model (specifying changes of presentation and navigation), and a presentation generator (to generate the results of the adaptation).

♦ **FAÇADE.** [Kur04] bridges the gap between Internet contents and heterogeneous computing environments by delivering web contents to mobile users. Context information captures the device's constraints and connection, and the user preferences. A distributed architecture separates context processing from content adaptation for ensuring flexibility and extensibility.

♦ **A Framework for Context-aware Adaptive User Interface Generation (CAAUIG).** proposes models for context, an architecture to support the adaptation lifecycle and a languages. It covers static and dynamic aspects of the context, user, platform, and environment information, a model-based approach starting from a task model, it uses rule specification language and user interface description languages [Han04].

♦ **SUPPLE.** is a framework-toolkit that treats the UI generation as an optimization problem. It takes into account the device's constraints and users' efforts. Its approach uses declarative descriptions of a UI, device characteristics, widgets, and a user and device specific cost function. According to Gajós *et al.* (2004) an adaptive UI requires 3 inputs: the UI specification, a device model and a user model. SUPPLE framework includes as input, a trace of typical user behavior, enabling user-specific renderings.

♦ **ResOurce-aware Application Migration (ROAM).** is an application framework that assists developers in implementing applications able to run in multiple devices, and that enables users to migrate their applications across devices without much efforts. This framework follows as adaptation strategies: transformations, dynamic instantiation and offloading computation. Agents are used to support the migration across devices. This framework considers as context information only the device properties, including: display size, input method and user interface library [Chu04].

♦ **Java Context Awareness Framework (JCAF).** aids domain specific context-aware applications with a Java API and a set of interfaces. It is distributed, service oriented, event based. It includes context services, access control, context client and monitor [Bar05].

♦ **Framework for Multimodal Adaptive Environments (FAME).** [Dua06] presents a conceptual framework that considers user, platform and environment information in order to adapt the presentation and the behavior of an interactive system. They

also propose behavioral matrix as the logic to define adaptation rules, an architecture and a set of guidelines.

♦ **XUL-based Interface Framework (XIF).** [But07] separates the UI adaptation from the logic to ease the development of mobile apps, assuring more portability for Java ME settings. XUL is cross-platform, widget based markup language based on existing standards.

♦ **PersonisAD.** is an architectural framework to model and to use context. Its key concern is scrutability, i.e. the users can access and understand their models by using simple operations like access, tell, and ask. Their main contribution is a generalized framework to simplify the creation of ubiquitous computing applications; they focused on modeling the environment and on the distributed and active nature of the models [Ass07].

♦ **Architectural Frameworks for Automated Content Adaptation to Mobile Devices (ACAMD).** Jankowska (2007) identifies as fundamental requirements for adaptation frameworks: the transformation of images and the identification of the delivery context. According to her, these are two common features that should be supported by each adaptation framework. The main goal of this framework is to enable cost-efficient development of mobile applications, by providing a markup language and an integrated development environment (IDE). The framework requirements include supporting: navigation, organization, image conversion, data integration, fragmentation, layout and style.

♦ **Context-Aware Workflow Execution Conceptual Framework (CAWE).** for Ardissono *et al.* (2007) and (2008) to enhance the flexibility of the workflow in web service composition systems, the context information and the adaptation rules must be explicitly represented in the adaptation logic. CAWE is a framework that manages context-aware applications with a hierarchical representation of the workflow, thus supporting the execution of alternative courses of actions and the context-aware invocation of web services. It considers the adaptation of the UI and the workflow execution. And as such, they believe it can be extended to handle complex adaptation rules.

♦ **Layered Context-aware Adaptation Framework (LCAAF).** [Pre09] proposes an integrated approach for context-aware adaptation associating the contents, the application, and a self-adaptive framework to the network. They focus on a large-scale network, distributed settings and context-aware adaptation in running applications.

♦ **A Fusion Framework for Multimodal Interactive Applications (FMIA).** [Men09] proposes a multimodal fusion framework for multimodal data fusion. Different devices provide data to be analyzed, interpreted and combined, aiming at enhancing the end user interactions.

♦ **MIMOSA.** [Mal10] focuses in mobile users and web-based services. This framework includes an architecture and a middleware to aggregate context from distributed sources. By coupling services the user preferences are detected and considered to choose appropriate adaptation policies.

♦ **Less Framework (LF).** is an adaptive CSS grid system for designing adaptive websites. Four layouts with 3 sets of typographic presets are available, all based on a single grid (Figure 11). The layouts take into account the platform type, namely: a default layout (of 992 pixels, for desktops, laptops, and tablets in landscape orientation), a tablet layout, mobile devices, and wide mobile layouts (for large mobile devices or smartphones in land-

scape orientation). The layouts vary in terms of amount of columns and margin widths [Kor12].

♦ **Conceptual Framework (CF).** [Fis12] defines a multidimensional framework for context-aware systems. Context awareness is a multidimensional goal whose further features, as adaptation, are needed to exploit contexts' full potentials. This work discusses the CAA pitfalls, challenges and trade-offs, in a conceptual approach (Figure 12).

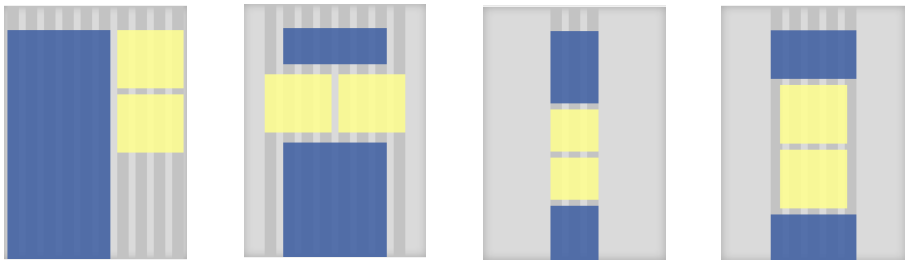


Figure 11. Less Framework 4: layouts for desktop, tablet, mobile devices, and large mobile devices. [Source: http://lessframework.com/]

The frameworks presented in this Section contributed with specific CAA concepts. Although they provide valuable contributions, they are constrained, either in terms of application aspects and domains or in terms of context. For instance [Dey01] and [Bar05] target at context information, but do not consider CAA. [Bar05] and [But07] are technology driven (to Java settings). [Llo97], [Per00], [W3C03], [Oli03], [Han04], [Dua06], [Ard07], [Ard08], [Mal10], and [Kor12] target at web-based applications. [Mal10] and [Kor12] focus on mobile applications. [W3C03], [Dua06] and [Fis12] by providing more conceptual solutions, cover also a more generic purpose, i.e. being domain-independent and targeting at multi-contexts.

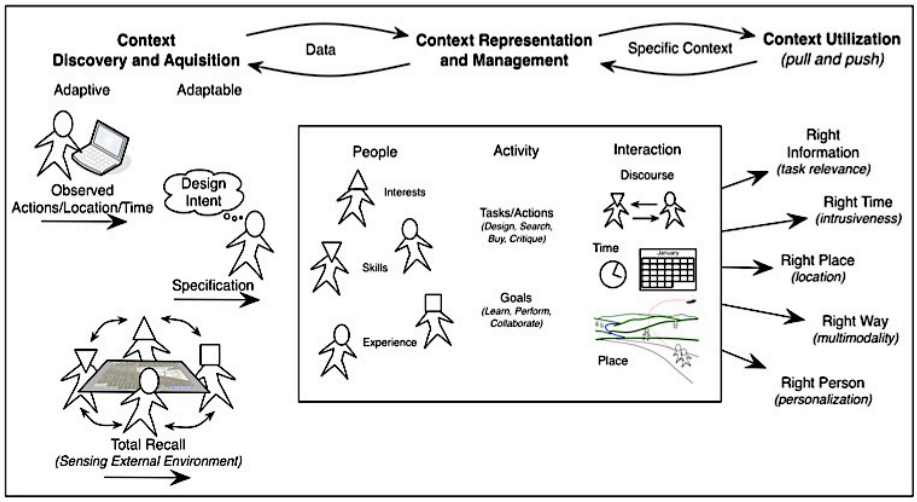


Figure 12. A multidimensional framework for context-aware systems [Fis12].

Our framework aims at extending this work by complementing the use of context information with appropriate techniques of adaptation. ROAM and Less Framework are limited in considering just the devices as context dimension. Oliveira and Fernandes (2003) work and PersonisAD are limited in terms of application domains, while the former focuses only in the educational domain, the later targets at ubiquitous computing. [Bar05] focuses mainly on medical applications.

PersonisAD and SUPPLE are limited in terms of the covered dimensions of the context, once the former considers just users and the later considers just users and devices,

but both exclude environmental characteristics. Moreover, often when only one dimension of context or aspects is considered, it is partially considered: e.g. Jankowska [2007] considers the platform as a context, but it is limited to mobile devices, besides just content adaptation is considered as applications aspect. The works of [W3C03] and PUC are also limited in terms of aspects, taking only the adaptation of the modality, i.e. presentation, into account. Less Framework takes just the screen dimension into account to adjust the only layout of graphical UI's [Kor12].

Table 4. Current frameworks according to the context dimensions (user, platform and environment), support provided and system aspects (presentation, navigation and content). From – non-existing, + low, ++ middle, to +++ high.

Related Works	Context			Support	Aspect		
	User	*Plat*	*Env*	*Type*	*Pres*	*Nav*	*Con*
FAHD [Llo97]	+	+	-	Framework, Architectural Forms	+	-	+
CFAWS [Per00]	+++	-	-	Algorithm, Methods	-	++	-
CaFT [Dey00]	+++	+++	+++	Conceptual Framework, Toolkit	+	-	+
PUC [Nic02]	-	++	-	Architecture, Specification, Language, UI Generator	++	-	-
W3C MIF [W3C03]	+	++	+	Generic Meta-Architecture	++	+	+
AEHS [Oli03]	+++	-	-	Decision Models, Analysis, Strategies	++	+++	+
FAÇADE [Kur04]	+	+++	-	Architecture, Decision Engine, Context Repository	++	++	++
CAAUIG [Han04]	++	++	++	Framework, Language	+	+	+
SUPPLE [Gaj06]	+	++	-	Framework-Toolkit	++	+	-
ROAM [Chu04]	-	+++	-	System, Architecture, Application Framework	++	+	-
JCAF [Bar05]	++	-	++	Service Structure, Program Framework	-	-	+
FAME [Dua06]	++	++	++	Conceptual Framework, Logic	++	++	-
XIF [But07]	-	+++	+	Logical Approach, XUL UI Framework	+++	-	-
Personis AD [Ass07]	+	+	+	Rule Language, Generic Architectural Framework	++	-	++
ACAMD [Jan07]	+	+++	+	IDE, Architecture	++	++	+++
CAWE [Ard08]	+++	++	+	Framework, Architecture, Logic	++	++	+
LCAAF [Pre09]	++	++	+	Design methodology	+	+	++
FMIA [Men09]	++	++	+	Fusion Framework	+	-	++
MIMOSA [Mal10]	++	+++	+	Architecture, Distributed Framework	++	++	+++
LF [Kor12]	-	+++	-	Layout Options	+++	-	-
CF [Fis12]	+++	+	++	Conceptual Framework	+++	+++	+++

Table 4 presents the 21 frameworks analyzed based on the context of use that they target (user, platform or environment), their main contributions (architectures, algorithms, languages, models, toolkits, etc.) and the main aspect subject to adaptation (presentation, navigation, and content). The dimensions were classified based on their impact, e.g. when several contextual information are considered, they were classified as '+++', and when few

information were (partially) taken into account, it was classified as '+'. When no information belonging to the dimension was considered (or reported), it is classified as '-'.

As also pointed by [Mal10], context information must be broadly considered, however as we can see in this table, most of the frameworks on CAA partially consider the context, i.e. rarely user, platform and environment are simultaneously taken into account [Dey00], [Ard07], [Ard08], [Men09], [Mal10]. For [Han04] common requirements for context-aware frameworks include: context gathering, model support, and data interpretations.

Still when the context is considered, the contextual information concerning the platform is prioritized instead of the user [Nic02], [W3C03]. Moreover, as points [Jan07] the complexity of the frameworks for adaptation (to access, understand, install, use, and apply them) leads to rejections of adaptation methods offered.

2.3.3 Design Spaces

Design Spaces define possible alternatives for developing applications regarding multiple dimensions. With an explicit representation of these options, a Design Space can be used before the implementation of a project, to present design's options, after the implementation to analyze and explore the alternatives and also to compare different projects.

Nigay and Coutaz (1993) define a design space for multimodal systems. When different modalities, as voice, gesture and textual are integrated, the user I/O in different times may vary [Nig93]. The design space deals with tasks at the granularity of commands, aims at identifying software implications and constraints during its development phases, and also enables classification. It considers concurrency and data fusion, and it includes as dimensions: modalities (sequential, and parallel), fusion (combined, independent), and abstraction level (meaning and no meaning). This classification space defines 4 classes of system for reference, characterization and reasoning concerning I/O properties of interactive systems. It enables to locate systems and to consistently compare them, being complemented by a software architecture.

Karagiannidis *et al.* (1996) define an adaptivity design space, as a set of pairs with temporal aspects and priorities assigned. This space characterizes adaptation based on determinants, constituents, goals and rules, it customizes requirements for different domains and user profiles, and attends to a generic-purpose. They organize adaptation's strategies in 4 main decisions: what to adapt (constituents), when to adapt (determinants, or UI states), why to adapt (goals) and how to adapt (rules) [Kar96].

Vanderdonckt *et al.* (2005) propose a design space for context-sensitive UI's (Figure 13). Containing adaptivity, adaptability, and context-awareness, axis including the target ('with respect to what'), agents ('who'), temporality ('when'), amount ('how many'), aspects ('what'), approach adopted ('with what'), and qualities ('for what'). This design space encompasses seven relevant dimensions for context-awareness [Van05] and also proposes a new method for developing context-aware UI's. It aids designers to locate, identify and separate events that change context and thus reconfigure the UI's.

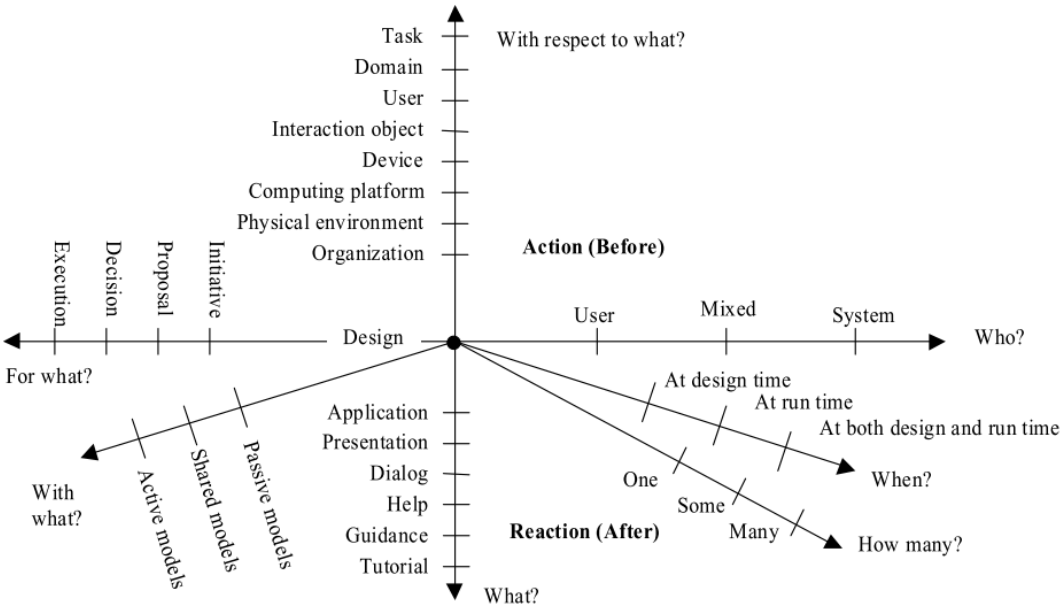


Figure 13. A Design Space for Context sensitivity [Van05].

Gajós *et al.* (2006) propose a design space for adaptive graphical user interfaces, analyzing aspects that affect the success of an adaptive UI. The associations among performance, accuracy, user satisfaction, cognitive complexity, adaptation's frequency and predictability were explored. As a result they noted that mechanical properties of an adaptive UI does not strongly affect the user's satisfaction or performance. Moreover users tend to prefer the UI's spatial stability. They also believe that frequent adaptations may reduce the utility of adaptive UI's [Gaj06].

Coutaz (2006) presents a dimension space that enables classification, comparison and contrasting different works on meta-UI's. Dimensions encompassed include: interaction techniques (integration, extensibility, representation, function) qualities (initiative and control) and functional coverage (services and object types). For each dimension, granularity levels have been defined, e.g. either the human or the system can take the initiative. The dimensions' levels are not necessarily exclusive or scalar. For Coutaz (2006), although various models and mechanisms are currently being developed for plastic and context-aware adaptive UI's, care must be taken to ensure that the end users still have enough UI control [Cou06].

For Calvary (2007), the problem space for adaptation can be structured in two hemispheres (left and right), integrating its lifecycle and its directives (rules). The lifecycle (Figure 14) is composed by 4 phases: definition, execution, evaluation, and capitalization. Each of these phases can be defined in terms of: who, what, where, when, and how. The rules associate conditions, reactions and events [Cal07].

Vanderdonckt *et al.* (2008) synthesizes the problem space for multimodal user interfaces (Figure 15). The following dimensions characterize this space: adaptation means (remolding and re-distributing), UI component granularity (from the interactor to the whole UI), the state recovery granularity (from action to session level), the UI deployment (static or dynamic), the context (user, platform or environment), the technological spaces (intra, inter or multi), and the existence of a meta-UI (none, without or with negotiation and plastic). These dimensions are scalar [Van08].

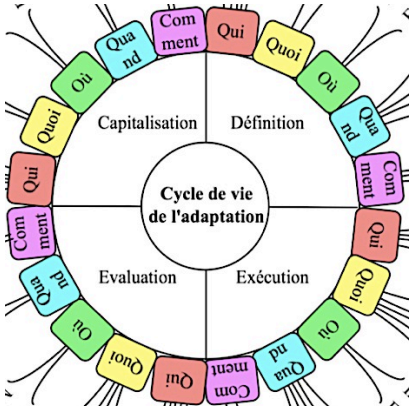


Figure 14. A partial representation of the hemispheres proposed by [Cal07]: a lifecycle of adaptation covering aspects as: who, what, where, when, and how.

Rouillard (2008) represented the main aspects of adaptation by means of a mind map. It includes: why, what, who, when, where, to what, and how. These aspects respectively cover: the goals of the adaptation, the system (navigation, presentation and features), the agent responsible for controlling the adaptation, the moment when it occurs (e.g. run time), the location where it occurs (e.g. internal to the system), the target context (user, platform, environment), and the approach (strategies, methods, etc.) [Rou08].

For Arhippainen (2009), the design space for adaptation includes as dimensions: target, means, and time. The target for adaptation refers to entities for which adaptation is intended: adaptation to users, adaptation to the environment and adaptation to the platform (i.e. physical devices and their characteristic). The means for adaptation denotes the software components of the system involved in adaptation. For instance, the system task model, the rendering techniques and the help subsystems can be modified to adapt to the targeted entities. Finally, the temporal dimension of adaptation refers to static adaptation (effective between sessions) or dynamic (at run time) [Arh09].

Cardoso and José (2009) focus on the gap between interactive features of displays and adaptation rules for contents. For them the fluidity and heterogeneity of social contexts must be considered, adapting the UI's design decisions. The user activity must be traced and used for the adaptation. The key problem is the diversity of interaction modalities and adaptation rules. So a design space informs designers of situated displays the relation among interaction modes, types of digital footprints they can generate and the adaptation they may support. Interaction options were analyzed and digital footprints categorized in: presence, presence self-exposure, content suggestion, and actionables. They define the mapping between interaction options and generation of local digital footprints. Adaptation types were analyzed and linked with each digital footprint [Car09].

Böhmer *et al.* (2010) define a design space for context-aware recommender systems to suggest mobile applications. Dimensions and techniques to capture information about the users, items and context were defined, as well as their relevance levels. Information captured both implicitly and explicitly has been considered [Böh10].

Calvary *et al.* (2011) define the problem space for multitarget user interfaces based on different constraints of the context and plasticity. This space (Figure 16) aids to understand and to reason about plasticity aspects. This space, called pStars, considers the UI as multivariate data including: domain-dependent UI's (interaction style and UI distribution), context (user, platform, environment), adaptation behavior (task, interaction style, occur-

rence, state recovery), and Adaptation control user interface (level of control, UI of adaptation).

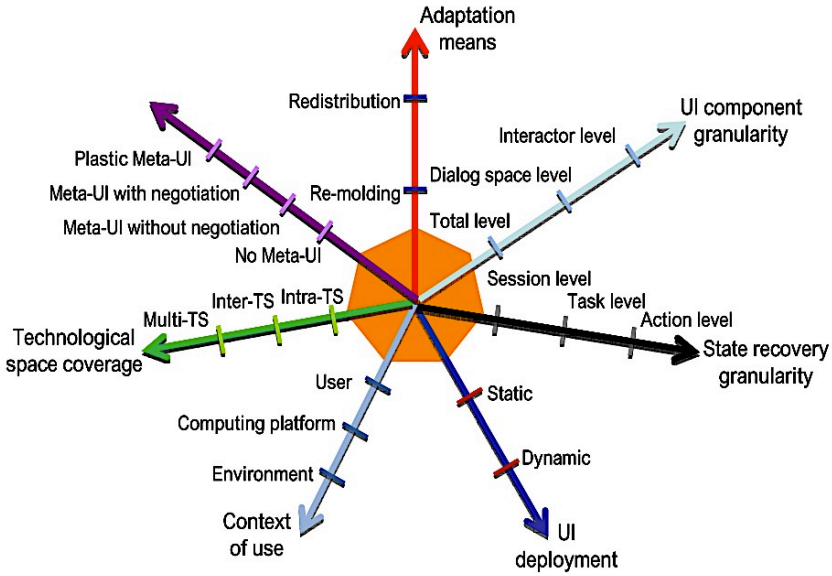


Figure 15. The problem space for plastic multimodal user interfaces [Van08].

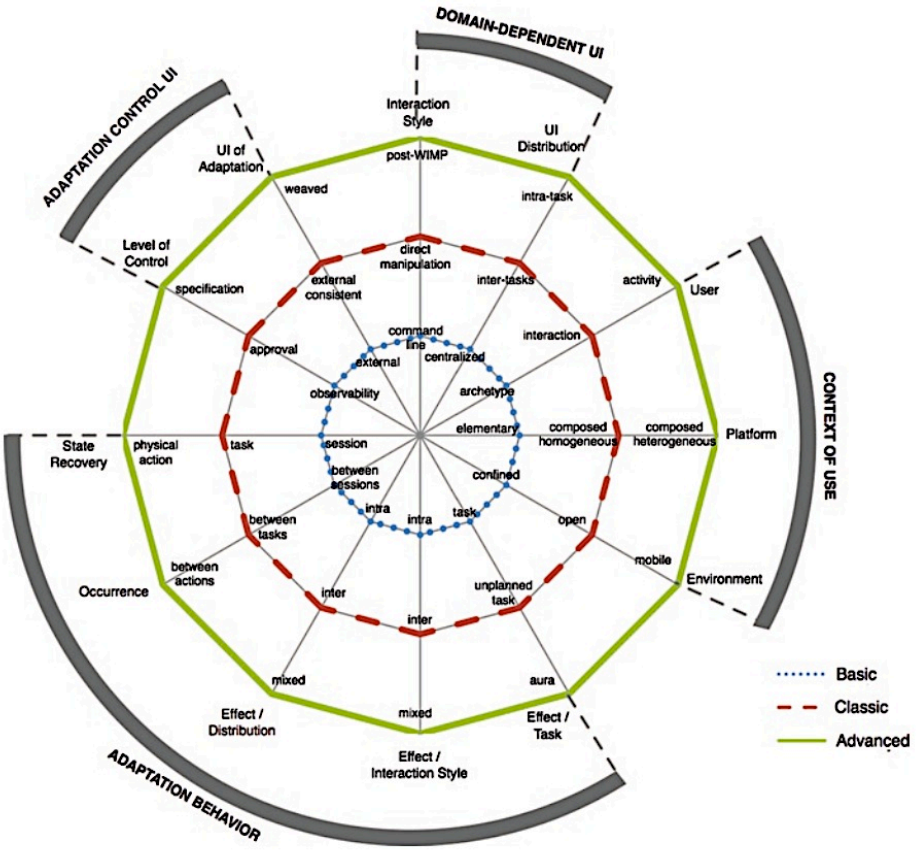


Figure 16. pStars a plasticity problem space [Cal11].

Table 5 presents a selection of 12 works that define design (and problem) spaces for adaptation of user interfaces or closely related concepts (as multimodal or context sensitive UI's). This table presents the dimensions and respective granularity levels covered in these works. Dimensions include discrete and continuous values, and their levels not always represent a scalar (ordered) relationship, although they are often represented by means of continuous axes.

Table 5. Existing design spaces, their dimensions and respective levels.

Design Spaces	Dimensions (levels' sample)
Nigay and Coutaz [Nig93]	Modalities (sequential, parallel) Fusion (combined, independent) Abstraction (meaning, no meaning)
Karagiannidis [Kar96]	What/constituents (semantics, syntatics, lexic) When/determinants (UI's states) Why/goals How/rules
SIMILAR DS [Van05]	To what (task, domain, user, platform, environment) Who (system, mixed, user) When (run-time, design-time) How many (one, some, many) What (application, presentation) With what (passive models, active models) For what (initiative, proposal)
Gajós *et al*. [Gaj06]	Costs x Benefits (low, moderate, high) Frequency x Predictability (less, most) Performance x Satisfaction
Coutaz [Cou07]	Technique (extensibility, design, representation, integration) Quality (initiative, control) Function (object types, generic services)
Calvary [Cal07]	Rules (condition, event, action, values) Lifecycle (definition, execution, evaluation, capitalization)
Arhippainen [Arh09]	Target (user, environment, platform) Means (navigation, content, presentation) Time (static, dynamic/run-time)
Vanderdonckt *et al.* [Van08]	Adaptation means (re-molding, re-distribution) UI component granularity (total, dialog, interactor) State recovery granularity (session, task, action) UI deployment (static, dynamic) Context of use (user, platform, environment) Technological space coverage (intra, inter, multi) Meta-UI (none, with or without negotiation, plastic)
Rouillard [Rou08]	Why (speed, simplification) What (feature, presentation, dialogue, help) Who (controller, agent) When (static, dynamic) Where (internal, external) To What (user, platform, environment) How (strategies, methods)
Cardoso and José [Car09]	Presence (detection, characterization, identification, exposure) Content Suggestion Actionables
Böhmer *et al.* [Boh10]	Users (implicit, explicit) Items (implicit, explicit) Context (implicit, explicit)
pStars [Cal11]	Domain-dependent UI (interaction style, UI distribution) Context of use (user, platform, environment) Adaptation behavior (task, interaction style, occurrence, state recovery) Adaptation control UI (level of control, UI of adaptation)

Analogously to the frameworks reported, the design spaces are dedicated to specific CAA aspects. While [Car09] tackles interactive displays, [Nig93] focus on multimodal applications, [Cou07], [Gaj06] and [Arh09] focus on quality and user experiences, [Rou08] focuses on multimodal applications and [Boh10] focuses on mobile applications. [Cal07] extensively covered aspects of the adaptation rules and its lifecycle. Although [Arh09] provides a precise definition, it is limited, i.e. it does not consider important aspects like what

is being adapted, or how, like [Kar96], [Van05] and [Rou08] do. [Van08] covers multiple aspects of a UI adaptation, however although the diagram represents scalar notation, it is not always possible to understand the dimensions as a set of ordered values, e.g. the context of use (user, platform and environments) per se cannot be analyzed with a scale (unless associated with quality levels). [Cal11] analyze plasticity and adaptation in an attempt to support decision making when designing advanced UI's.

Instantiating design spaces is challenging. When orthogonal axes are employed and a continuous line connects levels across dimensions, we tend to think that their scales are proportional. However such interpretation may lead to misunderstandings and incorrect conclusions, since the dimensions usually cannot be compared with numeric scales (e.g. context vs. adaptation means).

All dimensions identified with these related works are relevant to analyze coverage aspects and UI design. They are also complementary and establish a set of consistent criteria for analyzing UI's and interactive systems. However some shortcomings become evident (e.g. scalar representations for non scalar values, proportional associations for dimensions whose meanings are disconnected, narrowed focus).

2.4 Discussion

As presented in this chapter, the works dedicated to CAA are extensive. From an applied perspective, several domains can benefit from adaptation, as well as several information of the context can be considered to define it, and as several aspects of interactive systems can be subject to adaptation. From a support perspective, the contributions cover languages, models, approaches, methods and frameworks. The works analyzed particularly focus on: meta-models, frameworks and design spaces, due to these thesis goals. However such concepts often overlap, contributing in complementary dimensions of adaptation. From a temporal perspective for CAA, the ISATINE framework of [Lóp08] defines stages and gulfs for adaptation. It [Lóp08] includes 7 adaptation stages: goal, initiative, specification, application, transition, interpretation, and evaluation, linking 2 gulfs the execution and the evaluation (as Figure 17 illustrates).

The analysis of the state-of-the-art, although targeted at specific applied and supportive contributions, provided a large range of related concepts, including requirements for adaptation, implementations, techniques and lifecycle phases. The works that have been analyzed are often narrow in scope, excluding certain concepts, such as: context, system aspects, domains or the seven stages of Norman's theory of action [Nor86].

This theory has also been covered by ISATINE, and it describes how a user interacts with an application from the very beginning, when the user is forming his intention to reach a goal, until the end, when the results from the actions taken to achieve the goal are evaluated. The adaptation process should cover all seven stages of Norman's theory of action, including adaptation steps so important such as evaluation or the transition from the original system to the adapted one. Finally, the actors involved at each stage must be properly described to characterize any system with user adaptation facilities. In addition, how these actors collaborate to achieve on adaptation stage should be also considered.

Figure 18 illustrates the 7 phases of an adaptation lifecycle (according to Norman's theory of action [Nor86]) relating it to its 4 core components (adapter, context, models and rules) and respective reference information (who, to what, why, what, when, where, and

how) [Mot13]. At the top of the cycle, the adapter (who), i.e. the agent responsible for triggering or deciding the adaptation, has a goal and an intention in mind. Based on this information (why) and in what (to what) context information that surrounds the interaction moment, the system, by means of rules, can specify the actions that will change (how) one or more elements (what) of the interactive system, at design or run time (when), at the client, server or proxy (where). A transition can be then used to present the results of the adaptation in the models to the end user. Such results will finally be interpreted and evaluated by the agent (i.e. the end user, the system, or a third party). Depending on the evaluation given, the adaptation cycle can be concluded (if satisfactory results have been obtained), or continue (until satisfactory results are reached).

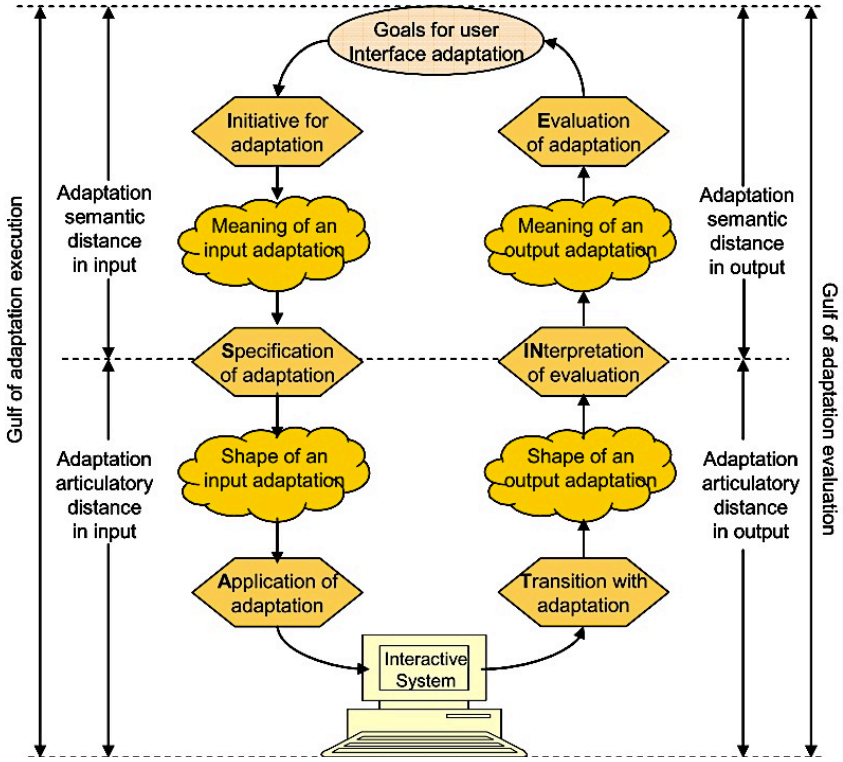


Figure 17. The ISATINE Framework: 2 adaptation gulfs (execution and evaluation) connect 7 stages of an adaptation process (goal, initiative, specification, application, transition, interpretation and evaluation) [Lóp08].

In this chapter we have explored a selection of works that cover both practical and theoretical perspectives of CAA. As *practical perspective*, application domains and aspects that can benefit of CAA (or that are subjected to it) have been presented. Finally we show how context information has been effectively considered for executing CAA. The application domains must be known to assure that our framework is sufficiently generic to accommodate different scenarios. The aspects define *what* specifically in an application can be subjected to the CAA, varying in different granularity levels, since a simple property until the complete application. The context information, as the most important unit for CAA, must be deeply investigated. It is relevant to know the variety of information that can be considered, and also its processes, e.g. how to gather it, treat, convert and prioritize. From a *theoretical perspective*, examples of CAA methods have been presented, including models, frameworks, and design spaces. Such works provided ground information to generate the TriPlet framework. The models are fundamental to define necessary concepts required for CAA, the frameworks support CAA different phases, e.g. its design, implementation, and evalua-

tion, and the design spaces define consistent criteria for assessing adaptation (analysis, evaluation and comparison).

The contributions of this thesis inherited a lot from works reported in this chapter, and this thesis attempts to: integrate, refine and extend them.

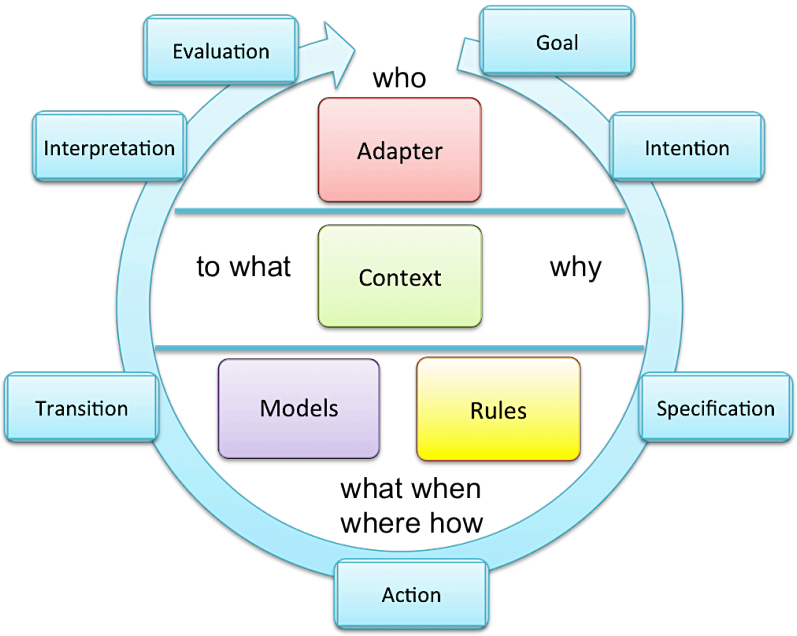


Figure 18. Adaptation lifecycle structured according to: its four core components (Adapter, Context, Models and Rules), seven reference information (who, to what, why, what, when, where, and how) [Mot13] and 7 phases (goal, intention, specification, action, transition, interpretation and evaluation) according to Norman's theory of action [Nor86] and also to ISATINE [Lóp08].

2.5 Shortcomings and Requirements

As presented in this chapter, several works have been dedicated to investigate CAA. However, normally they are constrained, focusing on deeply investigating: the context, specific system aspects, or one application domain. In this sense, the main shortcoming is the fact that the works are usually not integrated and often they do not consider a broad view of CAA. The analysis of the literature, as this chapter describes, revealed a set of shortcomings in the domain of CAA. Some of them receive special attention in the context of this project; they consist in limitations in current approaches, as for example:

S1. **Limited coverage of CAA dimensions.** Just one or two dimensions of the context information are usually considered at a time [Nic02], [W3C03], [Kor12]. Besides this, for each dimension only specific properties are taken into account (e.g., only the user profile or the device constraints), resulting in a limited context-awareness. The environment is rarely considered [Dey01], [Ard08];

S2. **Focused impact on CAA.** The system aspects targeted by the CAA are constrained, focusing mainly in the presentation, instead of navigation or contents [Gaj04], [Chu04];

S3. **Specific application domain.** Existing works are dedicated to specific application domains (e.g. educational [Oli03], medical [Bar05] and ubiquitous computing [Ass07]). Resulting in CAA approaches that are often unable to accommodate other domains.

S4. **Partial support for guiding CAA.** The current approaches do not support the complete development lifecycle, limiting the CAA process to context gathering [Dey01], [Bar05] or adaptation execution.

S5. **Moderate usability for CAA UI's.** The end user feedback is rarely considered to properly adjust the CAA process in a cyclic manner [Gan07]. Other qualities of service are prioritized instead [Lum02].

S6. **Technology-dependency of the solutions.** The technological space considered is limited, i.e. concerning specific languages (e.g. SMIL [Rou00], CSS [Kor12], [W3C03]), Java settings [Bar05], [But07] and platforms [Far07], [Kor12];

S7. **Obsolescence of the support provided.** The methodologies not always are extensible [Kor12] being often not synchronized with the latest technological achievements;

S8. **Simple logic to address CAA.** CAA is often tackled with simple rules [Gan07], while complex applications and contexts require more reasoning and inferences than these rules support;

S9. **Lack of unification of the vocabulary and approaches.** The current approaches are not unified and consistent, different names were associated to the same technique (e.g. action vs. transformation [Gan07], [Lóp09]), leading to ambiguous interpretations and misunderstandings;

The concerns and shortcomings observed for CAA delineate the problem space for this thesis. They also lead to conclude requirements and improvements in this domain considering different dimensions. The requirements identified so far state that:

R1. **Multidimensional context of use.** The context information cannot be limited to one, two or three dimensions, it must be not only broadly considered but complete and also enable extensions;

R2. **Multidimensional system aspects.** CAA cannot target to specific sub-properties of applications, the integral application, concerning navigation, presentation and contents, as well as their specific granularity levels must be carefully considered;

R3. **Application domain independency.** Once all application domains can benefit of CAA, theoretical methods that support CAA must be able to accommodate several scenarios;

R4. **Complete SDLC Coverage.** Methods must support adaptation in the entire lifecycle of development (considering, for instance, the feedback from the user to progressively adapt the application);

R5. **High Usability Levels.** End users must have highest priorities, being able to reject, accept, modify and evaluate CAA process. The adaptation engine must be able to evolve by learning with the end users;

R6. **Technology Independency.** The technological spaces cannot be constrained in terms of languages or platforms. The quick evolution of technology must be considered by enabling extensible and flexible approaches;

R7. **Extensible Approach.** The methods must be extensible allowing continuous update of concepts;

R8. **Advanced Logic.** Simple rules can be used as a basis for CAA, however more complex reasoning and inferences must be supported, e.g. with machine learning, ontologies, etc.;

R9. **Unified Vocabulary.** A standard terminology must be defined and largely adopted, resulting in more consistent approaches, enabling and facilitating the re-use and extensions.

Table 6 shows the requirements mentioned above related to the shortcomings identified in this domain. Although each requirement mainly addresses each shortcoming, they also impact in other aspects, contributing in multi-fold solutions. For instance, regarding the first and the second shortcomings (S1 and S2), the more context information covered (R1), the more system aspects are adapted (R2), as a benefit of the extensible approach employed (R7).

Table 6. Relationships between the shortcomings and the requirements of this thesis ('✔' mainly address and '•' contributes to address).

S / R	R1	R2	R3	R4	R5	R6	R7	R8	R9
S1	✔	•					•		
S2	•	✔					•		
S3	•	•	✔			•	•	•	
S4	•	•	•	✔	•	•	•	•	
S5	•	•		•	✔		•		
S6	•	•	•			✔	•		
S7	•	•	•			•	✔		
S8	•	•	•	•	•		•	✔	
S9									✔

The shortcomings identified lead to the definitions of the requirements that guided the problem space and the methodology of this thesis; the coverage of these requirements is discussed in the Chapter 5.

Chapter 3 TriPlet

The analysis of the literature review, as presented in Chapter 2, enabled us to identify fundamental concepts and main shortcomings in the domain of interest and lead to the consequent definition of corresponding requirements for this thesis. The methodology of this thesis as presented in the section 1.7 aims at tackling the shortcomings previously identified and consequently fulfilling the requirements mentioned in section 2.5.

This chapter presents TriPlet and its development steps. The literature review enabled the systematic extraction of fundamental concepts about adaptation. Then, TriPlet's components were created based on the analysis of the results of the systematic review. Fundamental concepts commonly found in adaptive and adaptable applications served as a ground for creating the CAMM meta-model, the techniques identified for adaptation were systematically organized in cards, and lead to CARF definition, i.e. not only adaptation techniques are needed to execute adaptation, so its principles, strategies and approaches were also considered. Finally, the CADS design space with essential dimensions for analyzing and comparing multiple adaptability levels of CAA was defined.

The second phase of the methodology of this thesis consists in elaborating a multi-dimensional CAA framework. This framework is based on the knowledge gathered during the systematic review. For [Han04] the many-fold requirements for systems with adaptive user interfaces include: handling both input and output using multiple mechanisms, supporting alternative metaphors, several platforms, and the automatically adapting interfaces to several different contexts of use.

The conceptual framework proposed in this thesis is named TriPlet, due to its three-fold contributions: a Context-aware Meta-Model (CAMM), a Context-aware Reference Framework (CARF), and a Context-aware Design Space (CADS).

3.1 Context-aware Meta-model (CAMM)

To formalize the essential concepts that are necessary to implement and execute CAA, a meta-model, named CAMM was created. CAMM is described in this section, but complementary details are reported in Appendix A and Appendix B. CAMM complies with the OMG (Object Management Group) notation for UML (Unified Modeling Language) Class diagrams: MetaObject Facility (MOF). *"The UML is a language for specifying, visualizing, constructing, and documenting the artifacts of software systems. The UML represents a collection of best engineering practices that have been proven successful in the modeling of large and complex systems"* [OMG00]. In this notation the associations are represented by named lines (e.g., triggers), aggregations represented by open diamonds (e.g., resource property), and compositions represented by closed diamonds (e.g., User). This meta-model besides covering the complete adaptation process, also abstracts fundamental concepts, establishes their relationships and defines their properties and methods. Moreover, additional information, as constraints and cardinality of the relationships, are also specified.

The CAMM was created based on the analysis of the systematic review results. CAMM inherits from several works reported in literature, mainly: [Koc00], [Kap02], [Fuc05], [Cal05], [Fah05], [Gan07], [Far07], [Lóp08] and [Mor12]. This section details the design decisions of CAMM.

Four colors were applied in the meta-model separating concepts that belong to different domains (Figure 19). Therefore, the classes represented by red blocks refer to the adaptation agents, the ones represented by green blocks refer to the context of use, the yellow blocks refer to the core of the adaptation process, and purple blocks refer to model generation. The enumerations that list potential instantiations of the concepts of CAMM are represented by blue classes.

This MOF-based[4] meta-model diagram, as Figure 20 illustrates, shows with the red blocks three possible agents to trigger an adaptation process: the system, the end user or a third party [Lóp08]. These agents are abstracted as 'Adapter'. Considering that an adaptation process may be composed by several phases, different agents can be responsible for each of them [Hor99] collaborating to defined the adaptations. For instance, the end user may start an adaptation process, and the system decides which is the most appropriate method among the available ones [Lóp08]. Besides, the agent roles can be further refined according to their specific characteristics and interrelationships, which permits collaboration and hierarchies. When a group of users is responsible for defining the adaptation, it is named crowd-sourced adaptation [Neb11b]. The advantage of such a collaboration is that different agents have access to different information, with priorities that may be more or less relevant depending on each scenario.

A CAA process can also be triggered by a change in the state of the context of use. The green blocks in the meta-model diagram represent concepts related to the context information. The context defines the adaptation rules since it provides information to instantiate them. For instance, when the user changes the orientation of the device, a technique like 'change the UI orientation' must be applied, rotating the content of the UI according to the new device position (information gathered for instance by a sensor). Changes concerning the user profile, the environmental characteristics and the application itself can all contribute to trigger an adaptation process.

As the context is a composition of information gathered from different dimensions, there are sets of rules that can be simultaneously applied. An adaptation process is then governed by one or more rule. Rules, represented in the meta-model diagram by the yellow blocks, can be syntactically structured in the form of ECA rules (event, condition and actions) [Dit95], [Sot07], [Gan07], [Lóp09], instantiated and triggered by context information. Due to the fact that more than one rule is often simultaneously applicable, conflicts may appear. In order to solve them, priorities must be assigned for certain contexts: adaptation techniques may be abstracted in policies (meta-rules) that can also be further abstracted as strategies (meta meta-rules). An extension of ECA rules that includes also Justification can be applied as well. Justification (J) provides a reasoning context for evaluations of ECA rules in order to support context dependent reasoning processes in dealing with uncertainties [Nge07].

CAA results can be presented to the end user with different methods aiming at preventing a potential end user disruption, such an issue is commonly caused by significant differences existing between the original UI and the adapted one, which can make users lost or confused. Animation is one potential solution that can be applied in this sense. By presenting the adaptation results by using animation, the intermediary steps of the adapta-

[4] MOF stands for MetaObject Facility, an OMG standard [OMG00] notation for meta-models (http://www.omg.org/mof/)

tion process are explicitly presented to the end user, a smooth transition from the original to the final UI may aid users to intuitively comprehend the changes in a sequential approach [Des11].

In a model-based approach. models for UI's are generated as consequences of the actions performed by the adaptation rules. In the CAMM meta-model diagram, the models are represented by purple blocks. In the model driven approach, following the principles of the CAMELEON reference framework, the models range from task and concept level, to abstract and concrete levels and then final level [Cal03], according to their abstraction level. While a task model specifies the tasks and subtasks involved to accomplish a specific user goal, the final UI level specifies the characteristics of the layout which in a GUI case correspond for instance to the style, the alignment, and the colors of the UI.

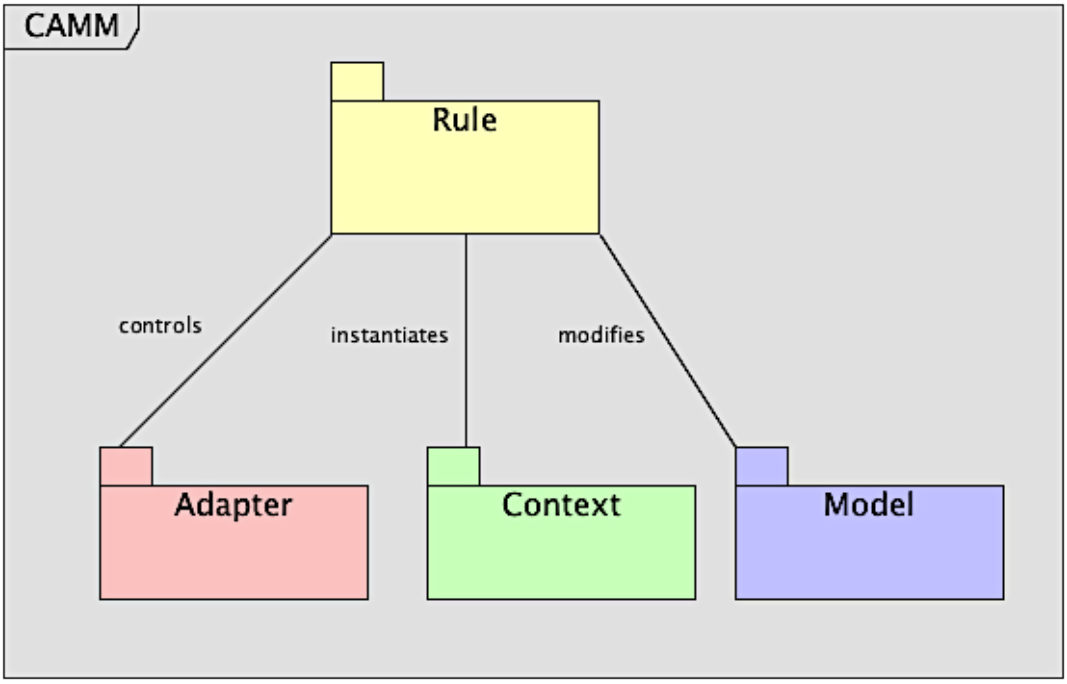


Figure 19. CAMM overview: 4 main packages.

3.1.1 CAMM: Descriptions of its 4 main concepts

The **Adapter** is the agent or the set of agents that is responsible for triggering or supporting the decisions for each of the adaptation phases [Koc01], [Sot07], for example: the end user that customizes the interactive system [Rou08], [Lóp08]. It has as *attributes*: id (the identifier of the adapter, a unique value), name (the name associated to the adapter), and priority (in a qualitative approach, it could be for example a value like low, medium, or high according to the priority associated to the adapter). It has as *methods*: `get()` and `set()` (generic functions used to retrieve the information about the adapters available in a given moment and to associate it to the attribute values, instantiating the adapter). It has as *relationships*: `inheritances` of User, System, and Third-Party, and `triggers` an AdaptationProcess.

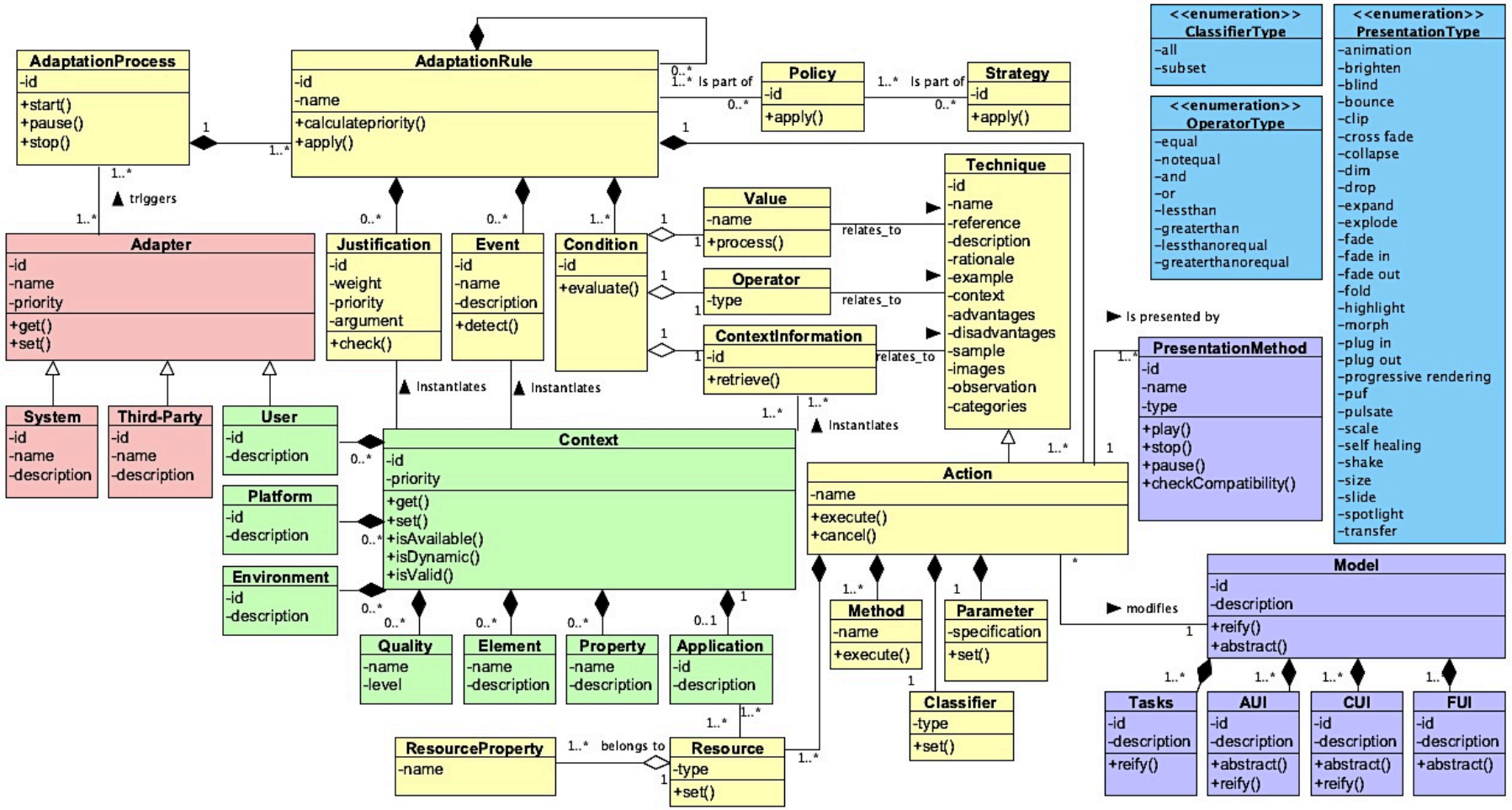


Figure 20. Context-Aware Meta-Model (CAMM). The corresponding descriptions are presented in Appendix A and its XML Schema is presented in Appendix B.

The **Context** is all the information that characterizes the context of use, the interaction scenario and that can be relevant and useful for defining the adaptation lifecycle [Dey00], [Kap02], [Fuc05], [Cal05], [Mor12] for example: the user John Doe interacting with a tablet PC in a train. It has as *attributes*: id (a unique identifier value for that context), and a priority value (if in a qualitative approach could be high, medium, low levels, this information is useful to solve potential conflicts between adaptation techniques). It has as *methods*: `get()` (to retrieve information about the context, coming for instance from sensors in the environment), `set()` (function to instantiate the values for the context, such values can be treated and processed beforehand if necessary, e.g. to convert units), `isAvailable()` (to check whether there is information to be retrieved), `isDynamic()` (to check whether the information varies along the time), `isValid()` (to check whether the information still holds). It has as *relationships*: `aggregations` of User, Platform, Environment and Application, and also Quality, Element, and Property. It instantiates a Justification, an Event and a ContextInformation.

The **Adaptation Process** is the set of all necessary steps to perform the adaptation, i.e. an adaptation lifecycle, for example: given a change in the orientation of the device screen (rotation), the layout of the UI elements change, and this change is smoothly presented to the end user. It has as *attributes*: id (a unique identifier associated to an adaptation process). It has as *methods*: `start()` (to begin the adaptation process), `pause()` (to temporarily terminate the process), and `stop()` (to terminate the process). It has as *relationships*: `is_triggered_by` an Adapter, `is_composed_by` one or more AdaptationRules [Fuc05], [Sot07], [Gan07], [Lóp09], [Lóp10].

The **Model** is a formal definition of an interactive system, that can be decomposed into different abstraction levels [Koc00], [Cal05], and complemented by different views, commonly expressed by means of a given notation (e.g. UML, XML, CTT), for example: a UsiXML model specifying an interactive system [Luy10]. It has as *attributes*: id (a unique identifier of the model) and a description (the model definition). It has as *methods*: `reify()` (a specialization of a model to make it more concrete) and `abstract()` (transformation to a higher abstraction level). It has as *relationships*: `is_composed_by` one or several models of Tasks, AUI, CUI and FUI and `is_modified_by` an Action.

3.1.2 Applying CAMM

During the SDLC a meta-model can be used before the development of an application, to guide stakeholders (developers, designers and project managers) to define the necessary requirements for the development phase. CAMM defines 4 conceptual modules (adapters, context, rules, and models) that must be considered by the development team before implementing an application. CAMM can also be used as a ground model to instantiate further documentation for a given project.

An adaptation process can be composed by one or several rules. Such rules can also be combined and abstracted, by means of policies and strategies; i.e. to leverage the reasoning when many adaptation rules apply or when the context of use is not deterministic, abstractions come into play. Policies and strategies represent higher levels of rules, in which the conditions of several rules are combined, for instance by means of priorities, and a set of actions, that are the most relevant ones for a given context, is defined as applicable. The relevance of the rules can be associated with costs (processing requirements, time) and im-

pact (significance and severity). Such approach aims at ensure that the rules are enough complete and correct to cover different compositions.

CAMM is composed by 34 classes, 72 attributes, 37 methods, 39 relationships, among which 10 associations, 4 aggregations, 21 compositions and 4 inheritances. There are also 3 enumerations (classifier, operator and presentation type) that complement the definitions.

The detailed description about such components is presented in Appendix A and the XML Schema is presented in Appendix B.

3.2 Context-aware Reference Framework (CARF)

For Hanumansetty (2004), the goal of a framework is to assist user interface designers and developers of interactive and ubiquitous software applications and to provide a basis for adaptive user interface for creating context sensitive user interfaces [Han04].

The **Context-aware Reference Framework** (CARF) is a reference framework created to list the most relevant concepts for implementing and executing CAA. The CARF, whose center is illustrated by Figure 21, is graphically represented by a mind map and composed by seven central branches. The mind-map, a type of spider map[5] [How05], has been chosen to represent CARF due to several reasons, including:

- it provides a unified view of an extensive list of concepts;
- it organizes and relates the concepts in a structured manner;
- it enables to navigate through the branches accessing contents in depth;
- the radial structure, rather than a hierarchical one, is close to the human approach to associate concepts, facilitating the comprehension and the understanding of the concepts and their relationships.

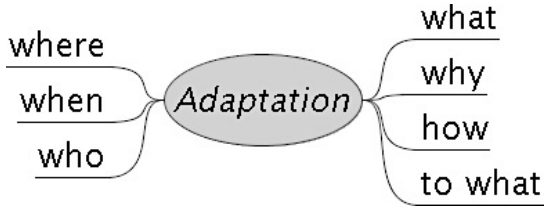


Figure 21. The central branches of the Context-aware Reference Framework (CARF).

While the central branches of CARF (i.e., the ones directly connected to the root) present abstract concepts (based on Quintilian's definitions [Qui12], as Figure 22 illustrates), the external branches (added under the central ones) list the possible instances for these abstract concepts, and thus aid the implementation, execution and analysis of CAA.

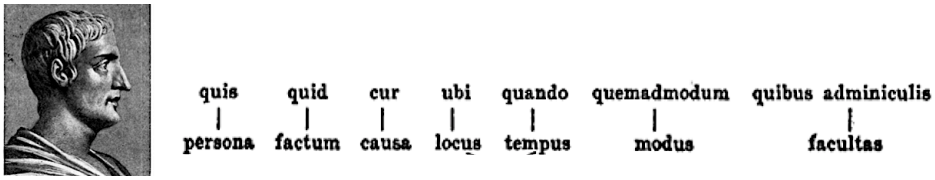


Figure 22. Quintilian defined 8 aspects (in Latin) that aid to investigate a topic, guiding the study and the understanding of it from different perspectives (who, what, why, when, how and by what means). [Source: http://www.phillwebb.net/history/ancient/Quintilian/Quintilian.htm]

[5] http://eprints.brighton.ac.uk/8159/

The definition of CARF inherits from several works reported in the literature, as: [Kar96], [Van05], [Cal07], however its core structure is closely related to the framework proposed by [Rou08], extending its definition. The evolution in the definition of the CARF is also documented in the D2.1.1[6] [Mot11] and in the D2.1.2[7] [Mot12].

To instantiate the CARF the following sentence must be appropriately respected and filled: At `<when>`, concerning `<to_what>`, the `<who>` `<where>` must `<how>` the `<what>` to improve the `<why>`. For example, in natural language, it could be: At *run time* concerning the *user visual impairments* the *system* at the *client* must *enhance* the *color contrast* to improve (or assure) its *accessibility*.

The seven central branches of the CARF represent, in clockwise sense, *what*, *why*, *how*, *to what*, *who, when*, and *where* dimensions, defined as follows:

- **What:** represents the resource type that is adapted [Ste95], [Van05], [Cal07], generally belonging to three main categories [Bru01], [Rou08]: navigational flow, presentation or content. For example: images;
- **Why:** defines the main goals for the adaptation process, which are expressed in terms of software qualities [Ste95], [Cal05], [Gaj06], [Rou08], [Lav10]. For example: adaptation can be performed targeting the improvements of the usability level;
- **How:** defines how the adaptation process is performed, by listing possible methods, techniques and strategies for the adaptation [Ste95], [Van05], [Cal07], [Rou08]. For example, with the adaptation technique of changing the font size;
- **To what:** lists context information that justifies and defines the adaptation process, i.e., usually the application resources are subject to adaptation according to the user, the platform, or the environment. For example: adapting according to color-blind users [Abo99], [Dey00], [Dey01], [Van05], [Zim07], [Rou08];
- **Who:** refers to the actor who triggers and is responsible for each phase of the adaptation process, e.g.: the end user, the system, or a third party. In a mixed approach both end users and system collaborate with the adaptation process [Van05], [Rou08], [Lóp09];
- **When:** represents the state in which the adaptation process is performed, i.e., it can occur at design time, run time, compilation time. For example: adaptations performed at run time [Van05], [Gan07], [Cal07], [Lóp09];
- **Where:** this branch refers to the 'location' in which the adaptation takes place, i.e., according to the architectural approach adopted it can be at the client, at the proxy, or at the server [Rou08], [Lóp09]. For example: adaptation performed at the server side.

These seven branches compose the core of the CARF, and by adding new instances they can be refined, however, according to our validation results (presented in Chapter 2, Chapter 4 and Chapter 5), they are enough complete to cover the most relevant concepts in this domain, being sufficient to comprise and to express all the necessary phases of an entirely CAA process.

The CARF defines the most relevant concepts for CAA and extensively lists and presents the possibilities regarding its implementation and execution. The CARF can be

[6] http://www.serenoa-fp7.eu/wp-content/uploads/2012/10/SERENOA_D2.1.1.pdf

[7] http://www.serenoa-fp7.eu/wp-content/uploads/2012/07/SERENOA_D2.1.2.pdf

used before the implementation phase of an application, as an extensive catalogue to guide developers in taking design decisions, or after the implementation phase of an application, to analyze and to evaluate the concepts that were considered, aiding also to identify under-explored areas for future extensions.

Given that the updated version of the CARF includes hundreds of concepts, to assure its readability, we list all its main concepts in the following sections, and keep the complete CARF diagram for the digital version, which is interactive, and permits to zoom in, out, to collapse and to expand the branches. More detailed descriptions are presented below and illustrated in the Appendixes: Appendix C, Appendix D, Appendix E, Appendix F, Appendix G, Appendix H, and Appendix I.

3.2.1 What

Many different resource types can be subject to adaptation. Generally three main groups of resources are defined [Bru01]:

- Content (each specific element that compose the interface of the application)
  - Audio
  - Image
  - Text
  - UI Elements
  - Video
- Navigation (the hierarchical structure of an application)
- Presentation (the interface with the end user, i.e. how the contents are arranged, structured and presented to the user in a given layout)

These concepts can be also refined with additional sub-concepts. The UI elements for instance include: textbox, combobox, radio buttons, forms, buttons, labels, etc. Furthermore the sub-concepts have specific properties that can also be targeted by the adaptation process.

3.2.2 Why

The branch *Why* of the CARF refers to the software quality that is aimed for the adaptation process. This concept is aligned with the definitions of non-functional requirements of Software Engineering methods, with the definitions of evaluation criteria for CAA as [Gaj08] and [Lav10] define, and it is also based on the [ISO9126]. In a high level overview, the main qualities considered include:

- Functionality
- Maintainability
- Portability
- Reliability
- Usability
- Consistency
- Extensibility
- Security

Each of the concepts mentioned above can be refined by related sub-concepts. For instance regarding the usability as a quality, the following sub-concepts are closely associated: accessibility, attractiveness, controllability, comprehensibility, error tolerance, feedback, flexibility, preview, learnability, memorability, satisfaction, control, effectiveness, efficiency, and collaboration. For Breiner *et al.* (2009) the two most important usability qualities for adaptive interfaces are: *memorability*, once the interaction is faster if users are able to remember and find items easily, and *familiarity,* once it is vital to match the interface to the real world. The *why* branch guides the adaptation definition, for instance the adaptation process can be oriented to improve the accessibility level of an application. It can be used before implementation (e.g., to orient the definition of the requirements), and after the implementation (e.g., in the evaluation phase).

3.2.3 How: Adaptation Techniques, Methods and Strategies

Adaptation techniques consist of atomic actions that are performed in application resources to modify them or their specific properties. Methods are a composition of adaptation techniques that are compatible and thus can be jointly applied to modify one or more resources of a given application.

In order to organize, list and describe several adaptation techniques and methods, in a systematic way, a card structure to detail them was created. This card (Figure 23) is composed of 11 information fields, which specifies for each technique: its name, description, references, rationale, example, context, advantages, disadvantages, a code sample (e.g., an algorithm), a picture (e.g., snapshot) and additional comments.

Figure 23 illustrates an example of such a template describing and illustrating the adaptation of the font size. Similar templates are being constantly filled and updated to achieve an expressive number of techniques gathered during the SLR phase. The contents are iteratively refined in order to keep its consistency (alternative names for the same technique must be identified and related, for instance) and to keep the references updated. A webpage (http://sites.uclouvain.be/mbui/caa/) is being continuously updated to organize, store, collaboratively maintain and extend the cards with adaptation techniques.

The adoption of a well-defined methodology permits constant updates once new adaptation techniques may rise, envisaging extensibility.

All the techniques and methods gathered so far by the CARF are listed in the Appendix E. They are organized according to their resource type.

While the adaptation techniques and methods compose the atomic units for an adaptation process, the adaptation strategies concern possible approaches applied to implement the adaptation process.

From a *technical perspective*, CAA strategies belong to two main categories:

- **Graceful Degradation:** occurs when the application is able to adapt itself according to the constraints imposed by the context of use, and thus reduce or degrade its functionalities or resources properties. For instance when scripts are disabled according to the browser type or version, and when videos are replaced by equivalent textual descriptions if the connection speed is lower than a given threshold.

- **Progressive Enhancement:** consists in providing additional options for the end user, in terms of functionality, contents or resources properties, according to her context of use. For instance, users with large screens are able to access more resources at a giv-

en moment. This characteristic of the context can be then appropriately explored by the adaptation (i.e. by optimizing the use of all end user resources and consequently providing them with a better experience).

Name	Font Resize
Reference	http://www.w3schools.com/css/tryit.asp?filename=trycss_font-size_px
Description	Change the font size according to the context.
Rationale	Given a text content, an adaptation rule is applied in order to change the text size, increasing or decreasing it.
Example	A visually impaired user accesses a news portal but the font size is inappropriate for reading. The font size can be increased, allowing the user to read the text.
Context	User with visual impairments, small, far screens, content in small sizes.
Advantages	The readability of the text will become possible or it will be improved.
Disad-vantages	The flow of the content may change, parts of the text may be hidden, scrolling may be required, the quality of the content may decrease (according to its resolution).

Sample (CSS)	`h1 {font-size:40px;}`
	`h2 {font-size:30px;}`
	`p {font-size:14px;}`

Picture	
Comments	The results of the adaptation must be evaluated once the information flow or content distribution may be affected.

Figure 23. Example of one CARF card.

From the *end user perspective*, the results of the CAA can be presented using different approaches. Once users may feel confused with disruptive changes between the original UI and the adapted one, animation can be applied to aid them to follow and to understand better such changes [Des11]. Although animations aim at aiding users to comprehend changes in a UI, they must be carefully designed, avoiding possible trade-offs, such as users overlooking the animation or even a significant performance decay [Dra11].

Some examples of animation approaches include: brighten, collapsing, cross fading, dim, expanding, fading in, fading out, morphing, progressive rendering, self healing, sliding, spotlighting, and plugging in and plugging out components. In CAMM, the enumeration presentation type lists 29 examples of existing animation approaches.

3.2.4 To What: Environment, Platform and User

This branch of CARF refers to the context information that is taken into account to orient and to define the CAA process. Context information involves all relevant information that is useful to implement an application, and it is usually classified concerning three main categories, namely: the environment, the platform, and the user.

The **environment** comprises the situation and the environment in which the interaction takes place [Bac10]. According to Zimmermann (2007) environmental differences result from the mobility of computing devices, applications and people, which leads to

highly dynamic computing environments [Zim07]. Unlike desktop applications, which rely on a carefully configured and largely static set of resources, ubiquitous computing, pervasive and nomadic applications are subject to changes in available resources such as network connectivity and input and output devices. Moreover, they are frequently required to cooperate spontaneously and opportunistically with previously unknown software services in order to accomplish tasks on behalf of users. Thus, the environment surrounding an application and its user is a major source to justify adaptation operations.

According to Coutaz and Rey (2002), the environment denotes the set of objects, persons and events that are peripheral to the current activity but that may have an impact on the system and/or users behavior [Cou02]. As such, the environment may encompass the entire world. In practice, the boundary can be defined by domain analysts who elicit the relevant entities for each case. Specific examples are: user's location, ambient sound, lighting or weather conditions, present networks, nearby objects, user's social networks, stress level.

Although commonly neglected, the context information concerning the characteristics of the environment in which the user is located is equally important. Relevant concepts include:

- Location
- External Events
- Presence and Arrangements
- Time

Each of the concepts mentioned above can also be refined. Concerning the presence and arrangements branch, the following sub-concepts are involved: entities, artifacts, natural objects, and people. Complementary concepts are illustrated at Appendix F. More knowledge about this context information, can be found in the works of [Zim07] and [Cou02].

The **platform** is modeled in terms of resources, which determine the way information is computed, transmitted, rendered, and manipulated by users. Examples of resources include memory size, network bandwidth, and input and output interaction devices. Resource characteristics motivate the choice for a set of input and output modalities and, for each modality, the amount of information made available [Cal02].

The platform branch certainly changes dynamically over time, and as such the CARF coverage considers also just the most relevant concepts regarding CAA. The items below illustrate these concepts:

- Interaction Modalities
- Operating Systems
- Input Devices
- Output Devices
- Browser
- Device Type
- Device Properties

These concepts can be also refined with related sub-concepts. The device type for instance include: desktop PC's, e-readers, netbooks, idTV's, laptops, mobile phones, note-

books, smartphones, tablet PC's, and PDA's. Additional aspects are illustrated in Appendix G.

Many works have been dedicated to define a concrete model for context information regarding the platform characteristics. Examples include: [W3C09] and [Ope12].

Concerning the **user** dimension, it is known that individual users differ on various dimensions, however, for most systems they still need to adjust their behavior and problem solving strategies to interact efficiently. The systems are, in general, designed for the average user, but not for all users. On the other hand, an ideal system should adapt itself according to the current users, thus compensating their weaknesses, providing appropriate help, and decreasing their mental and physical workload. Furthermore, an adaptive system should be able to characterize and distinguish individuals. User models support this task, by defining their knowledge, capabilities, preferences, cognitive strengths, and limitations [Nor89].

Context information regarding the user characteristics involves an extensive list of concepts. General examples include:

- Identification
- Interests and Preferences
- Educational Level
- Disabilities
- Profession
- Living
- Psychosocial State
- Socio-economic Class
- User State
- Interaction History / Usage Habits
- Health State
- Abilities
- Skills
- Demographics

For each of the concepts mentioned above, many sub-concepts can be associated. For instance the disabilities include: motor, cognitive, behavioral, visual, hearing, and speech. More detailed concepts are illustrated in Appendix H.

Since the CARF scope does not include retrieving all possible context information regarding the user, existing works already reported in the literature are considered to recover further context information that can be also relevant within the context of this thesis. Related works providing additional information include [Bac10], [UMO03], [Ver12].

3.2.5 Who

The branch *who* of the CARF defines the possible agents that trigger a CAA process, and who are responsible for its subsequent phases too. The main actors include:

- Developer
- End user

- System
- Third-party

It is worth to note that given the multiple phases that compose an adaptation process it is likely that multiple agents are involved collaborating in the decisions of adaptation. Besides, the actors involved permit the application to be classified as: Adaptable (when the user is responsible), Adaptive (when the system is responsible) or Mixed-Approach (when multiple entities are involved) [Lóp08], [Rou08].

3.2.6 When

The CAA can be performed in different phases of the development lifecycle process of an application, for instance during its design phase, during its compilation or on the fly, when the application is running or being executed [Rou08]. These phases are respectively defined as:

- Design time
- Compilation time
- Run time

3.2.7 Where

Different architectural approaches can be adopted to implement CAA depending on the complexity of the system [Sen13]. In the case of a client-server architecture a separate software client is developed for each device, which then communicates and interprets information from a dedicated, remote, server software application, in order to show a GUI [Mit07], [Mit09]. In this context, the location in which the adaptation is executed can be defined as:

- Client
- Proxy
- Server

Other examples of the proxy architectural approach can be found in [Bic99] and in [Buy00].

3.2.8 Applying CARF

The CARF can be used before implementing an application, as an extensive catalogue of possibilities to perform CAA. Besides, during the development lifecycle it can also be used to verify alternative options for implementation. And after the application was implemented, it can be applied to analyze the concepts that were previously considered, to which extent, and also to evaluate its coverage level and to compare multiple applications. By detecting the possibilities that were not initially considered, stakeholders are able to identify further opportunities for extending and updating their applications.

Although the CARF achieved a stable version concerning its branches, it is possible to extend and refine it with more instances, updating them according to the progress and evolution of new technologies, in an extensible manner. To do so the mind map file (available online at: http://sites.uclouvain.be/mbui/caa/) needs to be synchronized to accommodate additional instances.

The Context-Aware Reference Framework (CARF) provides an extensive list of concepts that can be used by stakeholders to propose, implement and analyze adaptive and adaptable applications. As such, the application of the CARF supports stakeholders during the complete development lifecycle of applications.

The coverage of the concepts considered in CARF is extensive (more than 150 adaptation techniques considered), however CARF is not exhaustive, so it also supports future refinements and extensions to be continuously updated according to the evolutions in the CAA domain in a scalable approach.

3.3 Context-aware Design Space (CADS)

The **Context-aware Design Space** supports stakeholders in the phases of implementation, analysis, and evaluation of adaptive and adaptable applications. The CADS aids developers before and after the implementation phases. Before it, they can identify possible dimensions and granularity levels for performing adaptation, and after it they are able to analyze, evaluate and compare these dimensions regarding their respective coverage levels. As such the CADS supports the analysis and the comparison of different applications that execute adaptation and during their complete development lifecycle. The CADS has been built in an iterative manner. First, relevant dimensions of CAA were identified based on the SLR. Then the specific granularity levels for each dimension were defined. In a first version of the diagram, due to the fact that not all dimensions represented ordered values, the interpretation of the diagram lead to wrong conclusions. As a result, the diagram was split. Dimensions that are not ordered belong to the CARF (for instance context information). Solely dimensions that have an ordered meaning were kept, for instance regarding the level of applicability of CAA (ranging from the entire application, to specific properties of the UI elements). The evolution of CADS is documented in the D2.1.1[8] [Mot11] and in the D2.1.2[9] [Mot12].

The definition of this design space inherited a lot from several work analyzed during the literature review. However the closest definition concerns the work reported in [Van08]. CADS can be considered as an evolved definition of the problem space defined in [Van08].

The CADS, as Figure 24 illustrates, is built as a radar chart, a useful approach to represent multi variable observations with an arbitrary number of variables. Although, in principle this representation is used for ordinal measurements, in the CADS case, qualitative values are represented with their respective empirical scale associated.

The CADS considers the benefits towards the virtues proposed by Lafon (2000) for design spaces, therefore it is *comparative* because multiple applications can be analyzed according to the same criteria; *exploratory*, because each dimensions can be analyzed in terms of exploration, i.e., identifying additional opportunities for extensions; and *descriptive*, because the dimensions are precisely pre-defined, in a consistent and unique way. Moreover, the CADS represents also an approach that is *extensible*, once dimensions can be added or better refined and *flexible*, once dimensions can be removed or added enabling focused analyses.

[8] http://www.serenoa-fp7.eu/wp-content/uploads/2012/10/SERENOA_D2.1.1.pdf

[9] http://www.serenoa-fp7.eu/wp-content/uploads/2012/07/SERENOA_D2.1.2.pdf

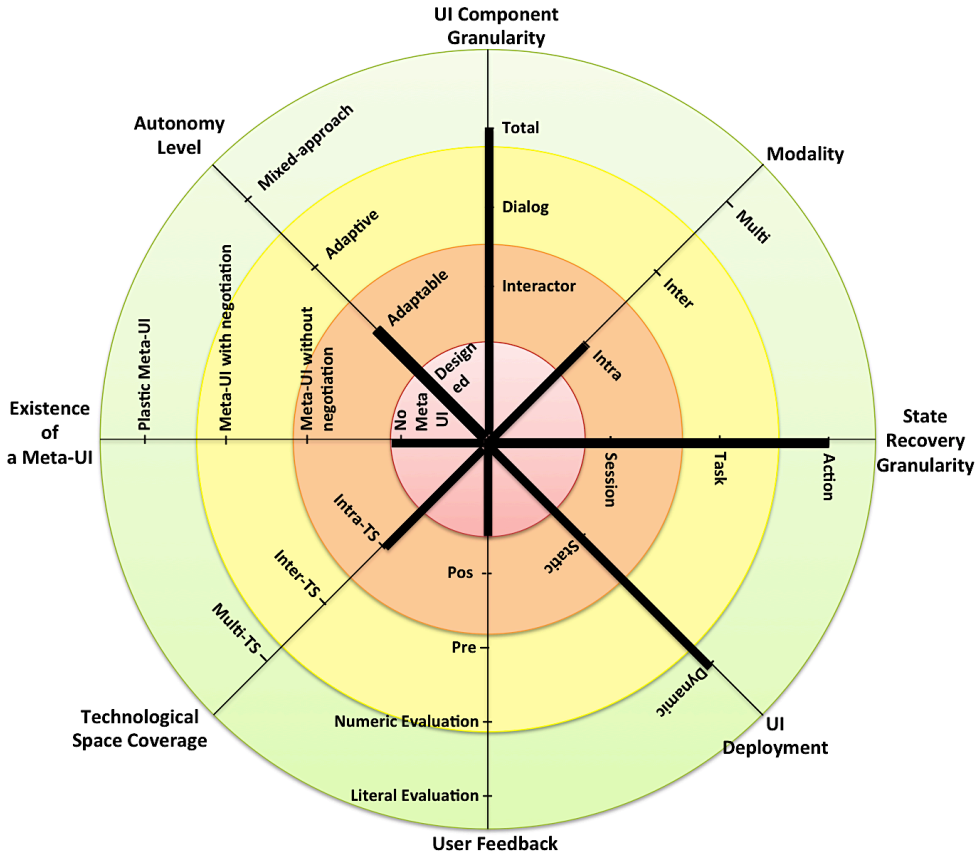


Figure 24. Context-Aware Design Space (CADS).

3.3.1 Reading and Interpreting CADS

Clearly, the interpretation of scales for the dimensions chosen can vary according to the context. However, it is a general interpretation that is provided by the CADS. Once the concepts cannot (in principle) be numerically evaluated and compared, it is their semantic meanings and interpretations that must be taken into account. The proportions are also empirically associated with the dimensions, because no formal experiments were conducted so far to identify real metrics that could be then associated with each dimension and its respective granularity levels. For each case of application of the CADS, its use must be defined and discussed. It is worth to note that all CADS dimensions, although comprised in the same representation, are still independent, and as such the concentric circles while aid the comparison of different granularity levels, do not necessarily represent the same coverage level between two different dimensions.

The central circle of the CADS, colored in red, represents the absence of adaptation concerns, for example when no adaptation process is performed an application can be classified as *designed* regarding its *autonomy level*. For each subsequent circle an additional coverage level of adaptation can be considered, and more external levels represent a higher coverage regarding one specific dimension for adaptation. So, for instance, an application able to adapt regarding many modalities (*multi*) can be classified as having a higher coverage level of adaptation regarding the modality dimension if compared with another application that performs adaptation within the same modality type (*intra*).

It is worth to note though, that a higher coverage level of adaptation regarding one specific dimension does not immediately imply a higher level of usability or a better appli-

cation for the end users. Implementing adaptation imposes many trade-offs (e.g., adapting an application may negatively affect its performance or accessibility level), and thus only by carefully planning and performing evaluation sessions, the actual benefits of adaptation for end users can be known.

The current version of the CADS results from the iteration between continuous evaluation and improvements, and thus it maintains the advantages of its previous version and discards potential issues that could lead to misunderstandings. This section explains the characteristics of the CADS, highlights its advantages and discusses its weaknesses.

As mentioned above the CADS diagram is extensible and flexible, therefore dimensions can be removed, inserted, or refined. Below there is a list of the dimensions included in the CADS as Figure 24 illustrates. Clearly, for analyses that require more focus, a specific set among these dimensions can be selected. On the other hand, for broader analysis it is also possible to include and consider further dimensions and granularity levels.

The eight dimensions described below in a clockwise manner represent the basic structure for the CADS. The sizes of the scales for each dimension level are arbitrarily defined (given that no numerical values or metrics have been associated):

- **User Interface Component Granularity:** defines the levels of abstraction for the UI elements that can be subject to adaptation [Van08]. Three levels are defined for these dimensions, interactor level, dialog level and total level. Interactors correspond to UI elements (e.g., a combobox), dialog refers to containers (i.e., a composition of UI elements), and total level refers to CAA that impact the complete window. It is worth to note that these examples are mainly applied in the context of GUI's and that the higher the level, the higher the impact that the end user will perceive, e.g. changing the combobox height has a lower impact than replacing it (concerning the end user perception).

- **Modality:** refers to the adaptations that change the modality type, i.e. the interaction techniques associated to a human sense [Van08], [Rou08]. During the user interaction, when the same modality is maintained after the adaptation is applied, the modality level is classified as intra-modality (e.g. when the volume of an audio content is lowered), when it changes from one type to another it is inter-modality (e.g. from audio to graphic), and when multiple modality types are involved and available, the adaptation is classified as multi-modality (e.g. users can access the contents in both audio and text simultaneously, instead of a video).

- **State Recovery Granularity:** refers to the application of the adaptation towards the impact in the continuity of the end user interaction, i.e., if the user is forced to quit the session and re-start a new one, the state recovery occurs at the session level, if the task is impacted the recovery occurs at the task level, and if just the action itself is impacted, the recovery is classified as at the action level [Van08]. For example, if the user is writing an email, each word typed represents an action, the task is the composition of the email, and the session corresponds to accessing his or her email box, logging in, and so on (thus including both task and action).

- **User Interface Deployment:** represents how much adaptation has been predefined at design-time vs. computed at runtime [Van08], [Rou08], thus respectively permitting a static or a dynamic deployment. CAA at design-time requires a new version of the application to be installed, while CAA at run-time corresponds to adaptations within the same application.

- **User Feedback:** refers to how the opinion of the user is taken into account, i.e., if the system is adapted, and the user can just accept or reject the adaptation after it has been performed, it can be classified as Post; if she is able to accept it (or reject) before it is applied, it is said to be Pre; evaluations refer to the possibility of the users to provide their feedback to the system, in a numeric (e.g., with a Likert scale) or literally (with qualitative values), providing additional details about their feedback. To gather the user feedback implicit, explicit or mixed approaches can be adopted, as discussed in [Voo13].

- **Technological Space Coverage:** is a working context with a set of associated concepts, body of knowledge, tools, required skills, and possibilities [Kur02], [Van08]. It refers to the technologies adopted and used by the application, when the same technology is maintained it is classified as intra-technological space (e.g. from a HTML document to another), when the technology changes between two different technological spaces, it is called inter (e.g. from a textual document in a *pdf* file to a video in *avi* format), and among multiple technologies, it is classified as a multi-technological space adaptation (e.g. from a text file in *pdf* to an animation with an audio file too).

- **Existence of a Meta-UI:** consists in abstract models that formally represent and handle the adaptation process and also allow users to control, evaluate and evolve it [Cou06], [Van08]. It comprises: no-meta UI, meta-UI without negotiation, meta-UI with negotiation, and plastic meta-UI.

- **Autonomy Levels:** refer to the extent in which adaptation is implemented, i.e., designed applications do not perform adaptation at all, adaptable applications rely on users to trigger and perform the adaptation, adaptive systems rely on the adaptation to be automatically performed, and self-modifying means evolutionary systems able to adapt their own adaptation engines [Cal07].

3.3.2 Instantiating CADS

To instantiate the CADS, the main axes are used to mark the coverage level for each of its dimensions. The extension of the marks is defined according to what the application provides in terms of adaptation. This permits a graphical visualization of the coverage level that is currently available. So, for instance if the adaptation regarding the UI component granularity occurs at the interactor level, the axis must be marked (highlighted) until this specific level. This procedure must be repeated for all dimensions. As a result the developer generates an applied CADS with easy identification of dimensions that were better explored and the ones that could be also taken into account to extend or improve the application adaptation in the future.

Another possibility to mark the dimensions consists in coloring (with darker tones) the region of the circle under the level of interest, however this approach works well only if all the levels correspond to the circles, and besides comparing multiple applications would not be possible with such an approach.

In order to compare two or more applications, developers have two possibilities: (i) parallel lines can be drawn in different colors, allowing a straightforward comparison; or (ii) an additional model can be used, comparing thus different application of the CADS in parallel. Both approaches permit multiple applications to be compared simultaneously, however for a large number of samples the second approach is recommended since it does not affect the readability of the dimensions' labels.

Once the comparison of multiple applications rely on color to differentiate them, it is necessary to choose different tones or styles, avoiding accessibility issues that may rise in case color blind users for example.

The Figure 24 illustrates the application of CADS to analyze a CAA case. In this example, the UI component granularity is classified as Total, the Modality is classified as intra, the State recovery granularity as Session, the UI deployment as Static, the User feedback as a numeric evaluation, the Technological Space Coverage as intra, the Existence of a meta-UI as meta-UI without negotiation, and the Autonomy level as Adaptable.

The CADS is versatile because it enables designers to analyze dimensions according to their own needs and interests, for instance by selecting the dimensions and levels they would like to consider. The use of the CADS diagram considering 6 given dimensions for instance permits developers to perform more focused analysis.

Besides this, the CADS is a flexible and extensible model, accommodating further dimensions and levels. The main criteria to perform this consists in assuring that it is possible to analyze the dimensions in a ordered way, for instance by defining different granularity levels, or a scale.

One example of refinement for the Autonomy Level dimension consists in adding a Mixed-Approach level on top of Adaptive. Mixed-Approaches occur when both the end user and the system are capable of taking decisions during the adaptation process.

The current version of the CADS is a result of the evolution of its previous versions; its weaknesses and strengths were analyzed and discussed during several project meetings of Serenoa, conference presentations (e.g. EICS and RCIS) and also with an internal survey consulting researchers and experts in this domain. Further details about the evolution process of CADS can be retrieved online at D2.1.2[10] [Mot11] and [Mot12]. The current version maintains the strong points of the preliminary versions and tries to overcome the misunderstandings caused by dimensions that are not ordered. To solve this issue, these dimensions were transferred to the CARF model, described in the Section 3.2.

It is certainly debatable which dimensions are the most important to be explicitly considered in the design space and one could argue in favor of adding or removing dimensions. In fact, we have tried to ensure that the selected dimensions consisted of a comprehensible and sensible set, by basing our choice on the following two principles. Firstly, there should be notable variability between applications with respect to dimensions. Secondly, a dimension should have a significant impact on the design and implementation of technical solutions [Rom04]. The following chapter provides examples to support such principles (Chapter 4).

It is worth to note that although the CADS establishes an empirical relation of order among the levels that compose each dimension, the concepts considered are still associated with qualitative variables, i.e., they (in principle) can not be numerically evaluated and proportionally compared. Therefore, for each application of the model it is necessary to justify the selection process and its respective application.

The main advantage of the CADS consists in the possibility to analyze adaptation in a unified graphical view, i.e., considering simultaneously a set of dimensions and levels that are relevant in the context.

[10] http://files.morfeo-project.org/serenoa/public/deliverables/M18/SERENOA_ D2.1.2.pdf

3.3.3 Applying CADS

CADS can be applied after the development of an application. It provides a tool for analysing, comparing and evaluating applications according to consistent criteria. Once instantiated, i.e. when an application is located within the diagram, the coverage levels of adaptation can be identified, as well as dimensions that have been underexplored and that could be more carefully considered in the next phases of development.

The CADS comprehends 8 dimensions and 4 ranges organizing 26 adaptability levels. It supports up to 3 application compared simultaneously in the same diagram.

3.4 TriPlet exemplified

The target audience of TriPlet components covers mainly UI designers and developers. Although TriPlet can be applied in the complete SDLC, there are specific phases that better benefit from its functionalities (Figure 25). The CARF can aid the project manager together with UI designers to translate the costumers' requirements into functional and non-functional requirements for the application envisaged. By means of the information provided by the CARF the stakeholders can identify adaptation techniques and further design decisions that contribute to fulfil the requests of the customers.

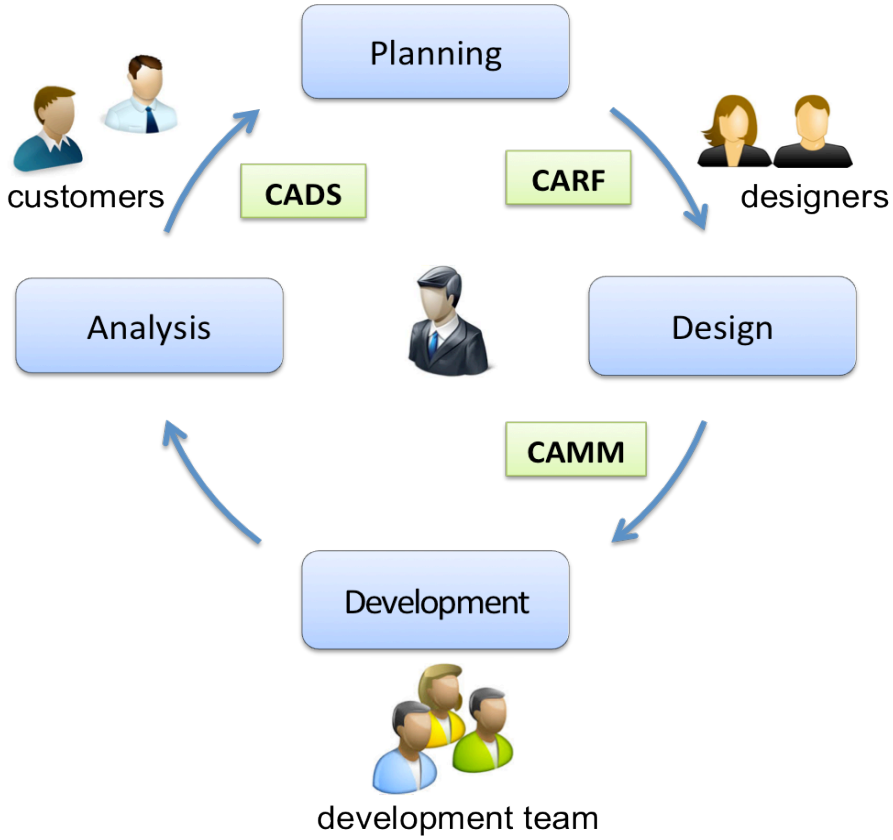


Figure 25. TriPlet, its target audience and the SDLC.

To translate the requirements into design decisions, the development team can rely on CAMM. By generating instantiations of the meta-model, according to the decisions defined by the UI designers, the application can be implemented. The definitions of CAMM provide ground concepts for developers, as well as the essential modules needed during the development phase.

Once the application is concluded, an analysis can be done by the project manager, together with the development team and UI designers. By instantiating the CADS, a given application is located and it can be presented to the customers as a tool to show how adaptation has been covered according to consistent criteria. CADS can also provide potential venues for extensions in the application analysed. In this activity, the project manager can use CADS to show to the costumers a comparison among different applications.

3.5 Final Remarks

This chapter presented the conceptual framework for CAA proposed in this project: TriPlet. This framework covers the complete SDLC for CAA and intends to be as generic as possible to support multiple application domains, contextual information, technologies, and design decisions. Targeting at fulfilling all the requirements defined in Section 2.5, TriPlet extensively considers CAA concepts.

To do so, CAMM is composed by 34 classes, 72 attributes, 37 methods, 39 relationships, among which there are 11 associations, 4 aggregations, 21 compositions and 4 inheritances (Figure 20). CAMM also includes 3 enumerations: classifier, operator and presentation type that complement the definitions with a list of 2, 8 and 29 examples, respectively.

CARF and its 7-dimensions core (Figure 21) comprehends 10 aspects types (*what*), 44 qualities (*why*), 152 adaptation techniques (*how*) detailed in templates as the one illustrated by Figure 23, 480 context information (*to what*), 4 agent roles (*who*), 3 stages (*when*), and 3 locations (*where*).

The CADS comprehends 8 dimensions and 4 ranges organizing 26 adaptability levels. It supports up to 3 application compared simultaneously in the same diagram (Figure 24).

The creation of the models of the computational framework is followed by a validation phase, i.e. defining case studies that benefit of CAA (selecting specific context information, selecting appropriate adaptation rules and techniques, and generating UI's that are adapted accordingly). To effectively validate the conceptual framework and its components, two application domains have been selected and three implementation versions were defined to instantiate TriPlet. They are presented and discussed in the Chapter 4.

Chapter 4 TriPlet Instantiation

In order to validate TriPlet, the conceptual framework proposed in this thesis, three different approaches have been combined in a more comprehensive approach. First, the literature was reviewed to search for and to identify relevant concepts in the domain of interest. This review also enabled us to verify the coverage of TriPlet support regarding alternative approaches and solutions for CAA. Then, two case study scenarios have been defined, and several instantiations of the framework were implemented, aiming to verify whether TriPlet is suitable for supporting CAA. Finally, the analysis of the previous approaches aided us to identify the lessons learned, in which a static analysis (ad hoc) has been performed based on a set of specific criteria. These approaches are detailed in this chapter. They are complementary and can be classified as:

- **Historical validation [Zel98]:** by collecting data from completed projects through literature search we identified the problem space for CAA, leading to a more comprehensive validation (i.e. focusing on context information, adaptation techniques and application domains);
- **Observational method [Zel98]:** through case studies valuable information could be obtained concerning the application of the TriPlet components, its feasibility and its potential benefits. Moreover it contributed to validate the protocol of usage (pipeline) of the framework and their limitations (lessons learned);

In the context of this thesis, to better explore in practice the potential of the conceptual contributions proposed by TriPlet (described in Chapter 3), and to illustrate its suitability, we selected two case study scenarios belonging to different domains. They illustrate two potential application scenarios: a *car rental* application and a *touristic* application.

To *rent a car*, end users, who are located in different contexts, specify: the period of interest (pick up and return dates), the car of interest (preferences), the locations of interest (pick up and return places) and their banking data to confirm the car rental. To *plan a touristic trip*, the end users, interacting from different platforms, access variable amount and type of information according to their context and request parameters, including: the location to visit, the weather information (temperature, clouds, wind, snow, rain), touristic sights to visit, and routes to follow.

The framework created, TriPlet, intends to be as generic, flexible and extensible as possible, therefore the design decision for implementing the case studies explore different alternatives of CAA, in terms of both: diversity of the context information considered and of the aspects that are subject to the CAA process. This chapter describes three scenarios of implementations of two distinct application domains. The development of the interactive systems described below all followed the definitions of TriPlet relying on the solutions provided by its components (CAMM, CARF, CADS and also the CARF Cards).

4.1 Specification of the Car Rental Case Study

The car rental example[11] considers a scenario in which the interactive system supports users in the task of renting a car. In this scenario, different context information can be used to adapt various system aspects, and to properly display the list of cars to rent, enabling users to make choices and to accomplish their main tasks with higher usability levels.

First, to formalize the definition and requirements of such scenario, a task tree, use cases, a domain model and a set of requirements have been defined. Then three versions of the application have been implemented considering specific contexts, aspects and based on TriPlet components. These implementations are described in the following sections. The choices of contexts aim at verifying a variety of combinations to explore TriPlet concerning its multidimensional focus (context information, system aspects, design decisions and technologies).

Figure 26 illustrates the hierarchical task analysis (HTA) [Kir92] for the car rental example. Basically, users must provide information about the car (i.e. category, color, model, and engine), then their own information (i.e. name, surname, address, city, ZIP code, country, gender, birthday, and email), and finally other information (i.e. commentaries, and maximum budget). Once the preferences are set and the request is submitted, users access the service and the results. To conclude the rental, users select a car, and define the period of interest. This task model is illustrative. By providing an abstract definition of the tasks, it serves as a basis for the implementations, leaving room for various CAAs that affect the tasks' sequence, so the tree still enables specializations and refinements.

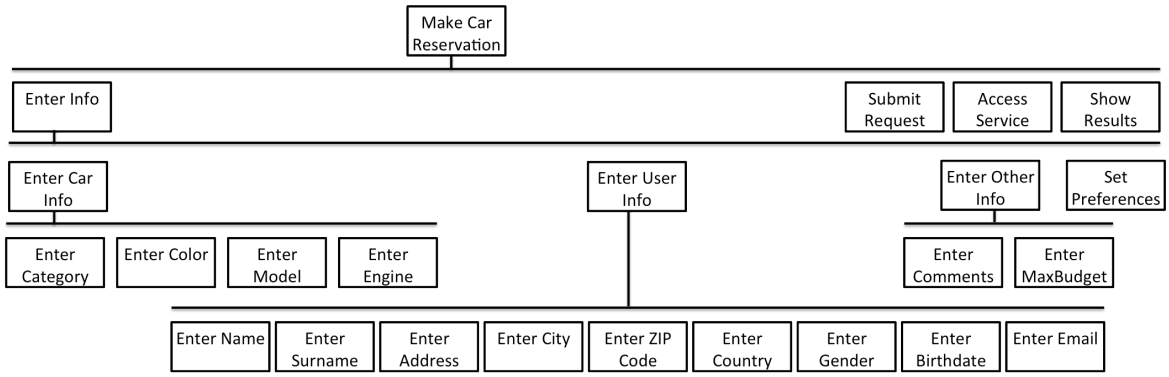


Figure 26. Hierarchical Task Analysis: Car Rental Example Task Model.

4.1.1 Domain Model

The domain model (Figure 27) specifies a basis for the data structure of the form fields, and defines which of the information is required. It defines the reservation of the customers in terms of date, place, payment and car. The class extra refers to the character-

[11] This case study has also been selected for the EU-funded FP7 Serenoa Project for demonstration. And for the W3C Working Group on Model-based User Interfaces (W3C WG MBUI) for validation. The implementations obtained in collaboration with the project partners is indicated when appropriate.

istics of the car of interest, such as air conditioning, gps or winter tires. This model aims at guiding the implementation by means of a consistent terminology for the applications.

4.1.2 Functional Requirements

A set of key functional requirements, based on the use cases provided in Figure 28, are considered for implementing the car rental example. They define that the users must be able to:

- select the city of interest to pick up the car;
- specify the period for the car rental;
- access a set of possible cars and select one;
- see details about the car of interest;
- access and select additional car features (e.g. GPS);
- provide personal information before renting the car;
- access details about the car rental before submitting the request;
- change the car rental parameters anytime before confirmation.

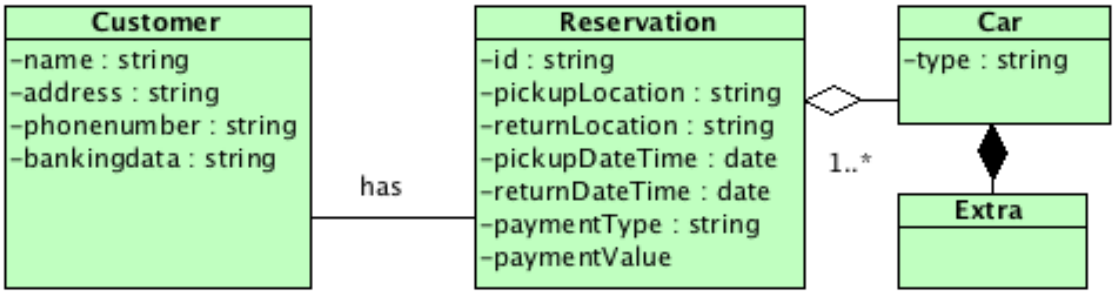


Figure 27. Domain model for the car rental example.

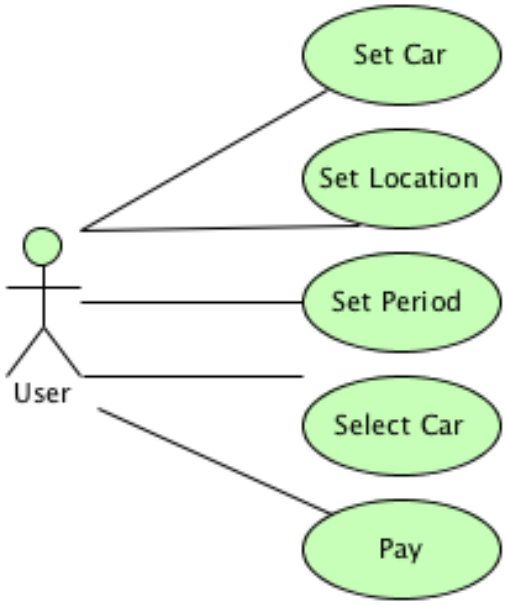


Figure 28. Use case diagram for the car rental example.

Based on the specifications provided by the documentation described, and also in the TriPlet components, three versions of the car rental example considering specific contexts of use were defined and implemented as instantiations of the TriPlet framework, as the following Sections detail.

4.2 First Implementation

The first implementation of the car rental example[12] takes into account the platform of the user, i.e. an Android tablet was adopted as platform, the level of expertise of the users (concerning their interaction with this type of system and domain), and the characteristics of the environment (level of stress). Two contexts of use were envisaged:

A. Users with low or no experience in car rental systems, medium experience in mobile applications, using a tablet device as a platform, and located in a calm environment (i.e. no loud noises, high stability levels, and sufficient free time);
B. Users with experience in car rental systems, and in mobile applications, using an Android tablet device as platform, located in a stressful environment and with a short time to conclude the task.

The design decisions concerning the specification of the platform are the same for both contexts, as such they can be considered for implementing the more abstract rules for CAA, i.e. meta-rules, mainly an Android tablet has a limited screen dimension and touch-based input controls. Android guidelines must be firstly respected, e.g. highlighting the selections in a touch-screen interface (thus providing immediate feedback of the users' actions). More specific CAA rules that consider the context of use A include:

- If the user is a beginner, then
  - all interaction steps and interaction status must be clearly indicated in the UI;
  - the amount of information displayed in the UI must be low (but sufficient), aiming to reduce the cognitive overload;
  - UI elements must be intuitive and simpler;
- If the environment is calm, then
  - detailed information about each interaction step can be provided, as well as additional features for interaction;
  - the main task can be split into several sub-tasks, and respective UI's dedicated to more specific actions;

[12] This implementation has been done thanks to the support of the master students Aldemar Reynaga Aramayo, Alexander Damnjanovich within the scope of the HCI course (LINF2356). A video file documenting the interaction steps is publicly available online at: http://youtu.be/M_7gu5FKFR0

Given that the task tree is affected by the adaptation, we illustrate two different versions of it, showing the adaptations performed in the 2 different contexts defined (A and B). Figure 29 and Figure 30, illustrate the two adapted trees that have been generated based on the original version presented in Figure 26. These trees have been adapted aiming to correspond to the constraints imposed by the contexts of use in scenarios A and B, i.e. users in a relaxed situation and with low level of expertize in the domain can have more detailed information and the task split in several sub-tasks that are more specific, while more experienced users in a stressful environment must have higher performances, thus they have interaction tasks that can be quickly concluded.

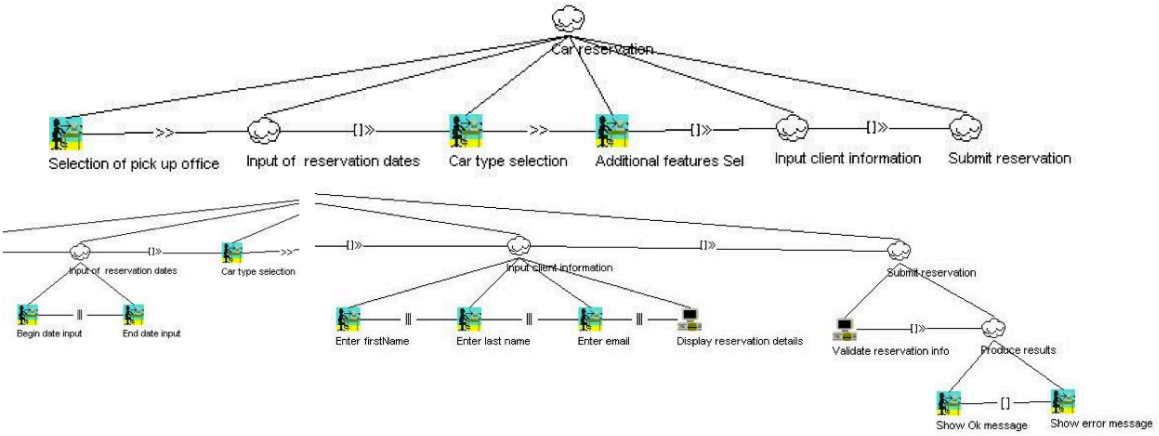


Figure 29. Task tree adapted for context of use A.

According to the definitions of the adapted task trees (Figure 29 and Figure 30), the respective UI's have been generated. They support users in performing the car rental tasks in the two scenarios envisaged. The Figure 31 and Figure 32 illustrate respectively such UI's. While in the first 4 interaction steps were envisaged, in the second just 2 interaction steps are required to accomplish the main task (rent a car). The second UI also provides auto-completing features, e.g. the calendar for the period and the possible office locations.

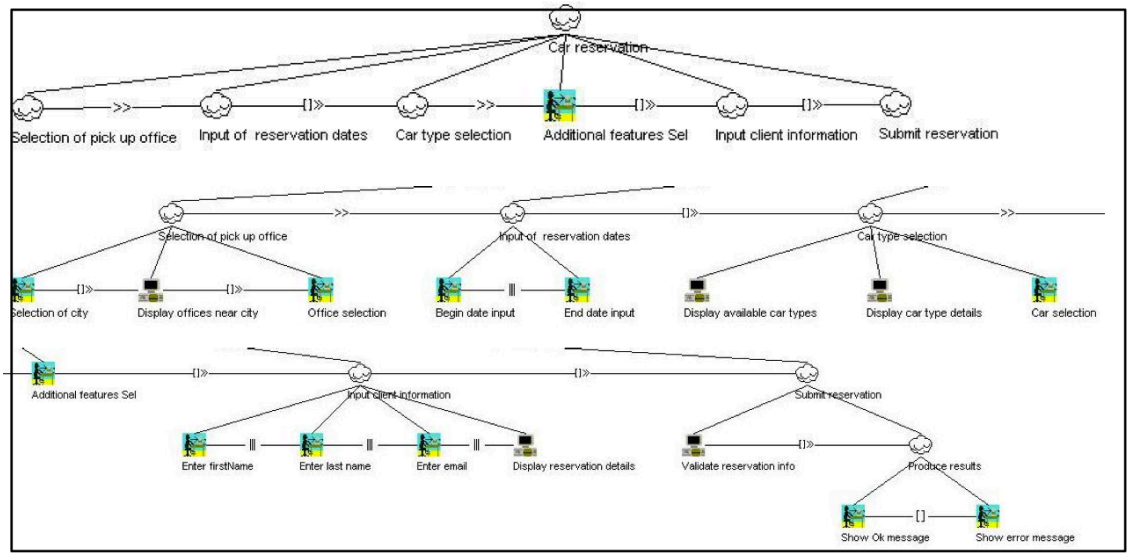


Figure 30. Task tree adapted for context of use B.

Although these features aim at improving the end users' performance (in case of experienced users), they could also cause cognitive overload for beginners or make them confused or lost, thus in these examples just users with time constraints (in a hurry or stressful environments) and high experience levels interact with these features.

The contexts covered by these examples combine different information coming from users, platforms and environments, and the systems aspects that are subject to the adaptation involve the contents, the presentation and the navigation. The classes of CAMM are instantiated in these examples in the following manner:

- As context, the users with high and low level of experience in the application domain (mobile applications for car rental), the platforms cover tablet devices with Android operating system, the environments with or without a stress level (users with time constraints, and high performance needed).
- The adaptation rules cover the contexts of use described above and change the contents of the UI (e.g. UI elements: the calendar with or without auto-complete features), the presentation of the UI's (layout, distribution and composition of the interfaces concerning the amount of components available in the UI and the level of details for each task), and the navigation (the structure of the tasks and their organization change to better suit the different scenarios of use).
- As models mainly the task tree and the final user interfaces have been affected, although also abstract and concrete UI's have been defined.
- As agent mainly the system is responsible for controlling the adaptation process.

In the scenarios proposed we assume that the abovementioned adaptation rules apply, significantly affecting the performance and the satisfaction of the end users. For a more accurate validation of such assumptions, further user studies may be necessary in order to effectively proof whether the qualities are impacted in the expected manner.

The Figure 33 and the Figure 34 respectively present how this first example is located within the CARF and the CADS.

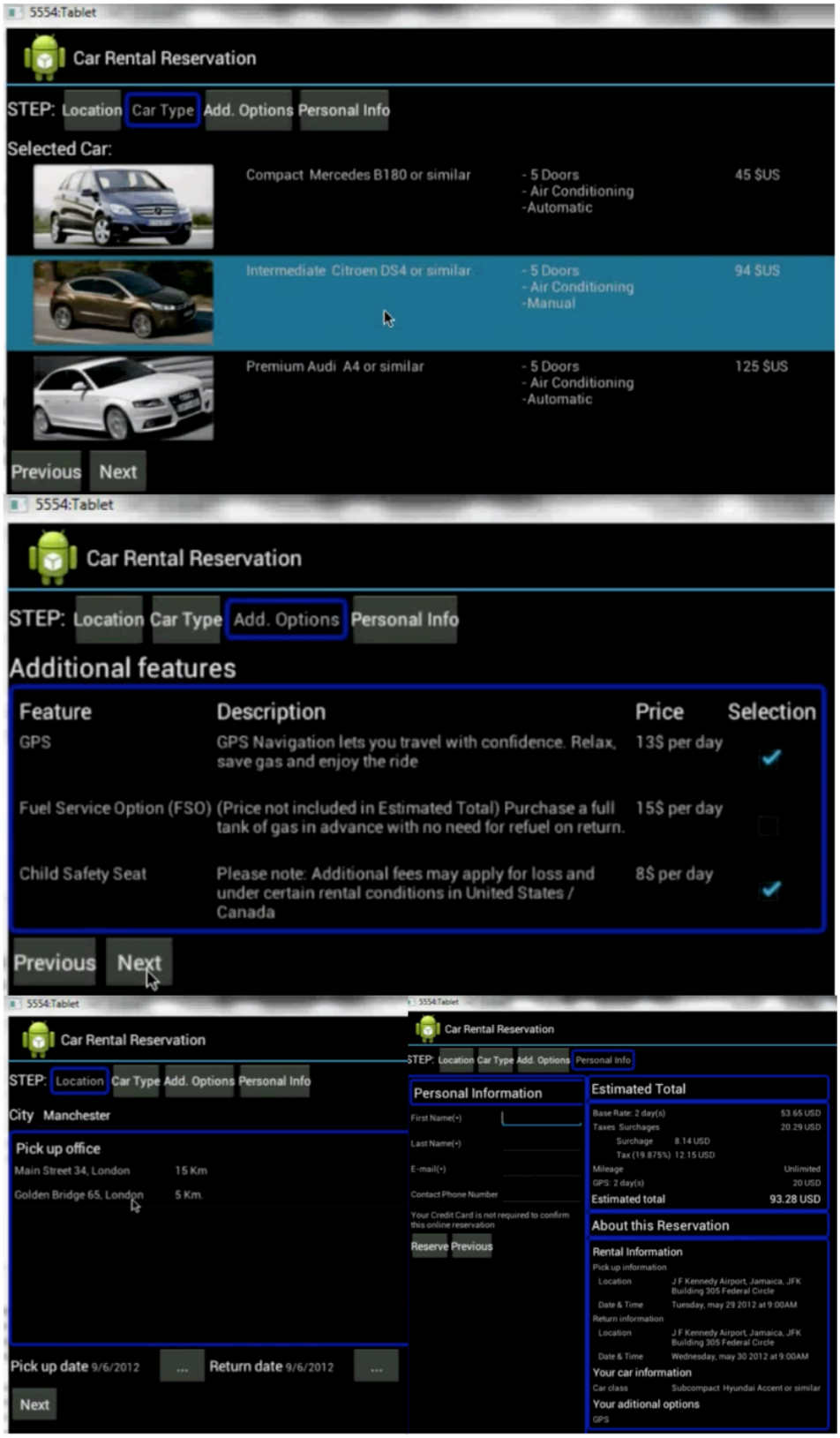


Figure 31. First implementation of the car rental example showing 3 UI's for Android tablet. In the context A: 4 interaction steps are available (Location, Car Type, Set Options, Personal Info), the UI's intend to be simpler and more intuitive.

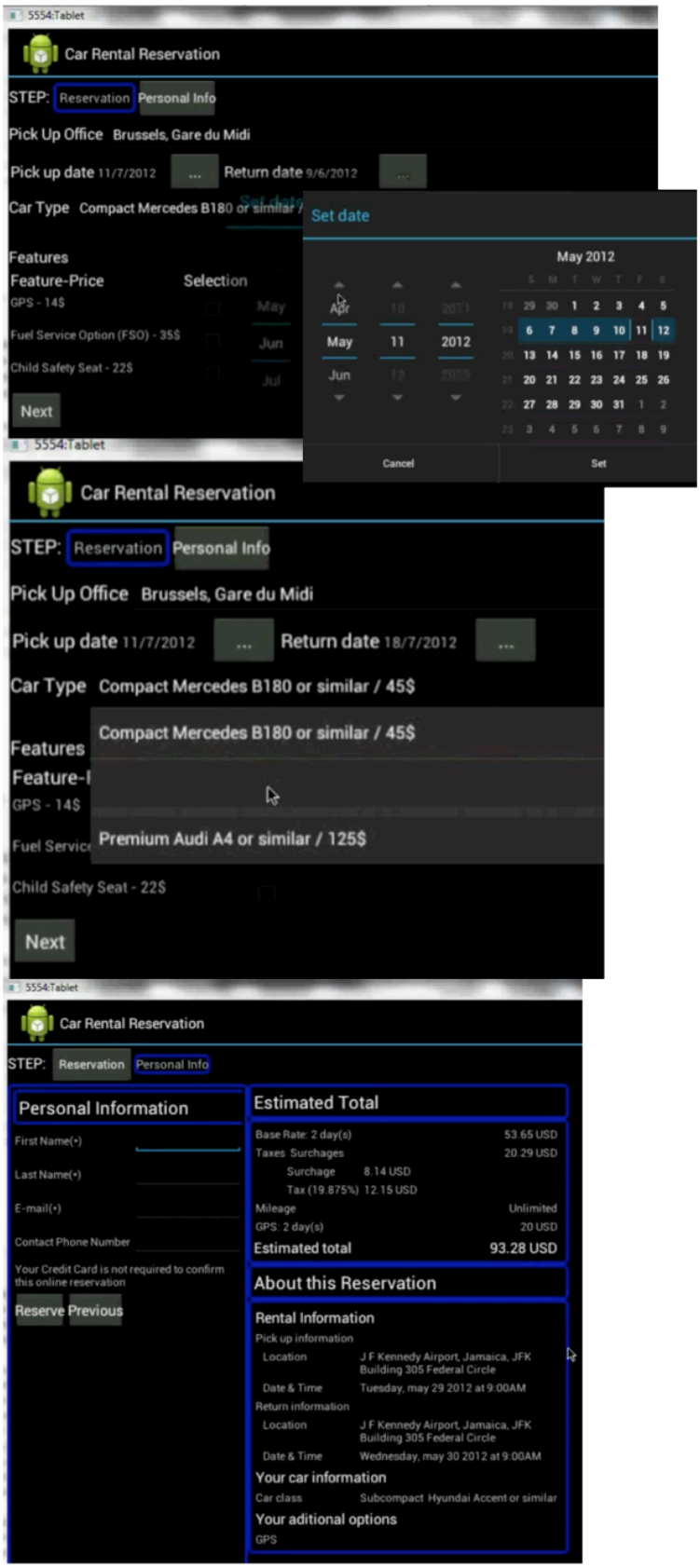


Figure 32. First implementation of the car rental example. 3 UI's for the Android tablet. In the context B: 2 interaction steps are available (Reservation, Personal Info), users have auto-complete features included to select the date and the car.

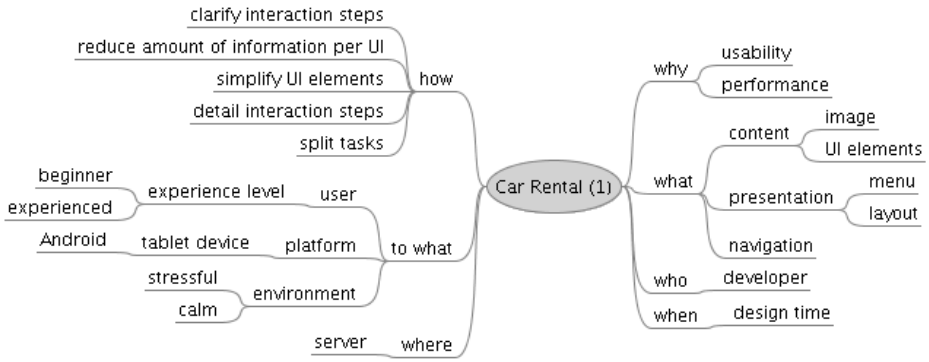


Figure 33. Instantiation of the CARF for the first implementation of the car rental example.

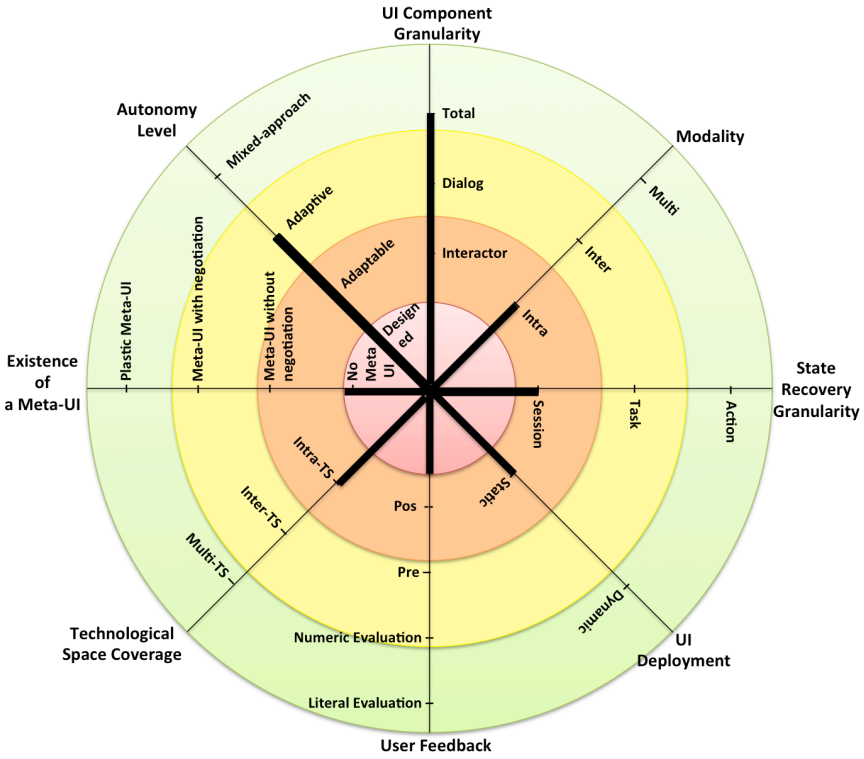


Figure 34. Instantiation of the CADS for the first car rental example.

These instantiations illustrated cover the two scenarios of the car rental example.

4.3 Second Implementation

For the second implementation[13] the specificities in the context for the car rental example take into account the screen dimensions and resolution. The layout of the web

13 This implementation has been done thanks to the support of the master students Thibault Goemans, Michael Lacroix, Grégory Nuyttens, Sébastien Scoumanne within the scope of the HCI course (LINF2356). The video is available at: http://www.youtube.com/watch?v=0iwGhhpuQCs

page is adapted automatically and progressively to fit the contents in all space available and therefore minimize scrolling.

jQuery Masonry[14] is a plugin of jQuery that arrange the UI components according to the re-size of the browser. Each UI component is treated individually, and moves to another column (or row) of the layout to fit accordingly in the new browser window size. Thresholds are used to assure the balance of the layout, avoiding unnecessary scrolling. The drawback of this solution is that developers must organize the components of the page in logical units. Once it is done, the re-organization is automatically and progressively performed.

Any screen dimension can be considered, because the fine-grained adjustment of the layout is done based on the progressive re-sizing of the browser. Three types of adaptation techniques were adopted to compose the CAA rules:

- **Resizing elements:** scaling font size, UI elements as videos and images;
- **Reorganizing elements:** changing the position of the components horizontally and vertically to assure a balanced layout;
- **Mixed approach:** a combination of resizing and reorganizing.

The car rental example comprehends three main interaction tasks: first users authenticate themselves, then they provide personal information, and finally they select the car and period of the rental. To enable users to accomplish these three tasks, seven logical units were defined (Figure 35): personal information, address, car type, car specification, period of the rental, additional specifications, and comments. Figure 35 illustrates 3 adaptation examples, in A a horizontally-oriented alignment is displayed, e.g. for a super-wide screen all UI components can be co-located, in B a balanced UI layout is presented, both horizontal and vertical alignments are considered, and in C a vertical alignment is considered, i.e. the UI components are displayed one above the other. Once only the screen properties and the UI components (size and position) are considered in this example, further rules with more specific CAA can be adopted extending it.

The instantiation of the conditions of the CAA rules vary proportionally according to the re-size of the browser window, i.e. the bigger the window, the bigger the UI elements, and amount of columns and rows of layout.

The contexts covered by this second example combine information from users and platforms, and the systems aspects that are subject to the adaptation involve the contents, the presentation and the navigation. The classes of CAMM are instantiated in these examples in the following manner:

- As context, the preferences of the users are considered, and also the space available in the screen size of the platforms used, more precisely users preferences regarding the dimensions of the browser windows and the screen size are used as context information.

14 http://masonry.desandro.com/index.html

♦ The adaptation rules cover the contexts of use described above and change the contents of the UI (e.g. re-sizing the font of the text), the presentations of the UI (e.g. layout of the UI units and the elements included, and the composition of the interfaces concerning the arrangements of the UI components), and the navigation (the structure of the tasks within a UI unit may change to better suit the different scenarios of use).

♦ As models mainly the task tree (reflected in the definition of the UI units and their internal components) and the final user interfaces have been affected.

♦ As agent just the developer is responsible for defining the thresholds for the adaptation, and the UI units (i.e. the organization of the features available per chunk) and the end user is responsible for controlling the adaptation concerning the dimension of the browser window.

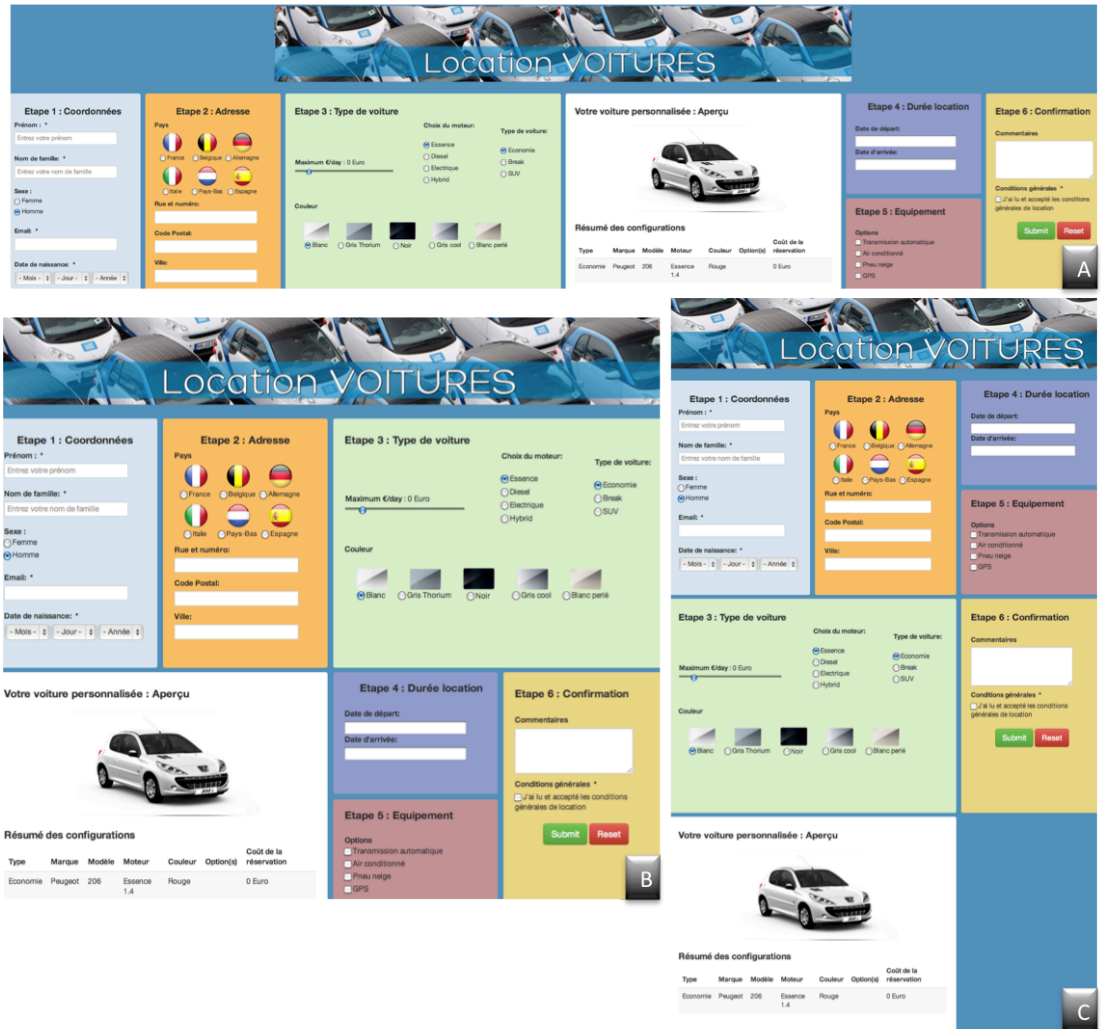


Figure 35. Second implementation of the Car rental website adapted examples: (A) Horizontally-aligned (e.g. for super wide screens); (B) Balanced Layout (e.g. for a tablet pc); (C) Vertically aligned (e.g. for vertically oriented screens).

The instantiations illustrated in Figure 35 cover the three scenarios of example (horizontal orientation, balanced layout and vertical orientation).

The Figures below present how this second example is located within the CARF (Figure 36) and the CADS (Figure 37).

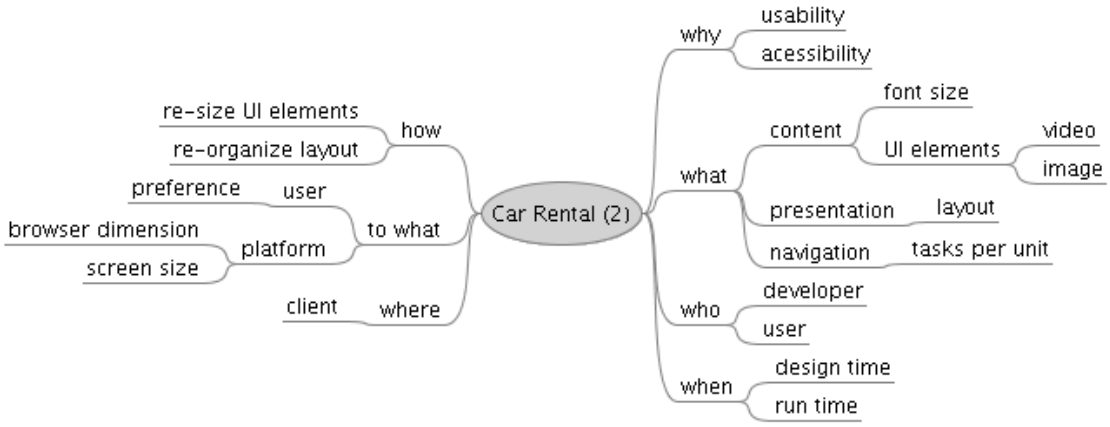


Figure 36. Instantiation of the CARF for the second implementation of the car rental example.

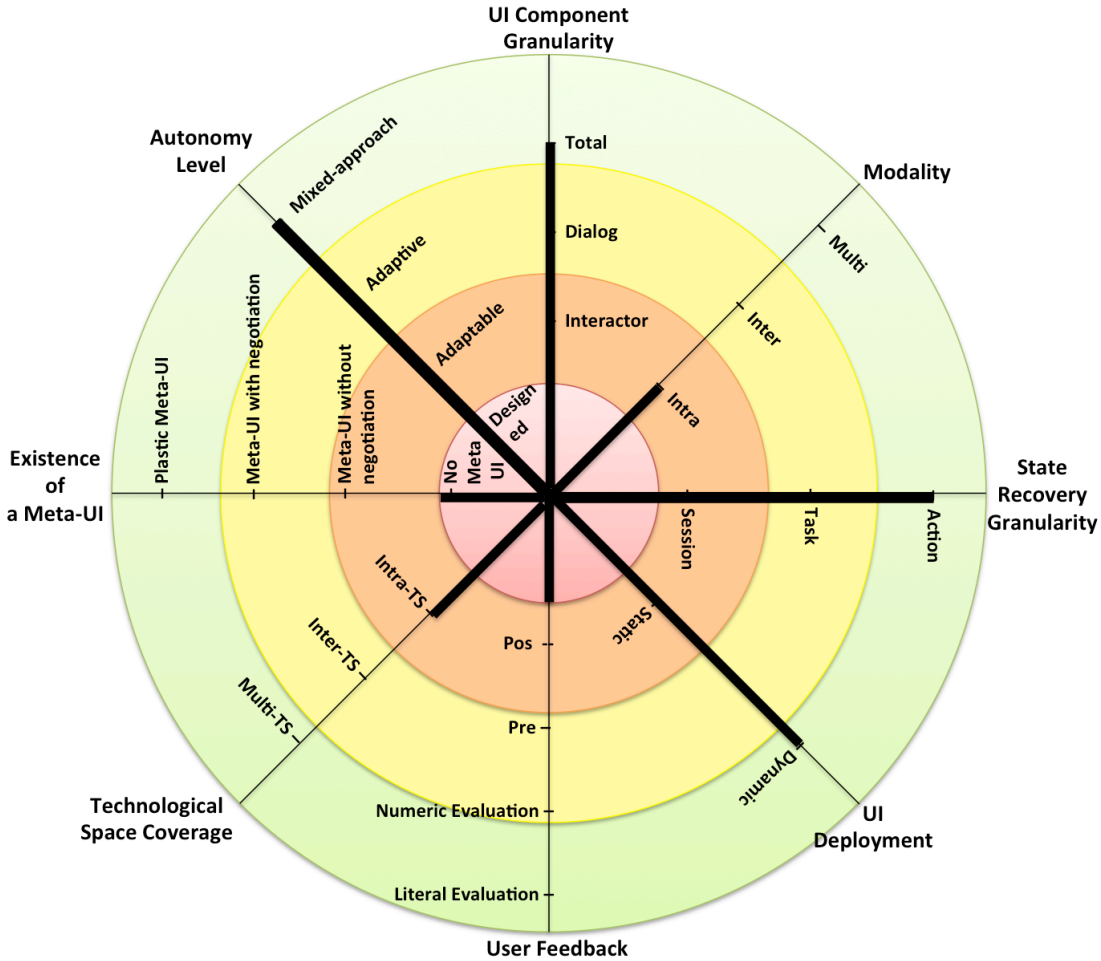


Figure 37. Instantiation of the CADS for the second car rental example.

4.4 Third Implementation

The car rental example was also applied in a third scenario[15] of CAA, performing adaptation techniques according to: the user visual impairments (color blindness), the platform type (mobile phone, tablet device), the battery level of the device, and the user preferences (set in the system).

Six adaptation techniques were chosen and implemented (e.g.: changing the modality and the image colors), aiming at assuring good usability and accessibility levels, by adapting the presentation (e.g. menu elements), and the content (images and text). The CAA was collaboratively decided by: the user, the system and the developer, and it was executed in the server during both: run time and design time.

Figure 38 illustrates the 7 adapted UI's for the car rental application, in 1 the adaptation is performed according to the user visual impairment (color-blindness), in 2 according to the dynamic device capabilities (battery charge level) and in 3 static capabilities of the device (an iPad tablet or a smart phone iPhone).

The contexts covered by this third example combine information from users and platforms, and the systems aspects that are subject to the adaptation involve the contents, the presentation and the navigation. The classes of CAMM are instantiated in these examples in the following manner:

- As context, the visual impairments of the users are considered (colour-blindness), and also the type of device and the level of battery charge of the platforms are covered.
- The adaptation rules cover the contexts of use described above and change the contents of the UI (e.g. change the colour map of the images if the user is colour-blind and replacing the video by an image if the charge level of the battery is low), the presentations of the UI (e.g. layout of the UI according to the space available on the device screens), and the navigation (the structure of the tasks and the UI arrangements may also change to better suit the different screen dimensions).
- As models mainly the task tree (reflected in the definition of the UI layouts) and the final user interfaces have been affected.
- As agent the developer is responsible for defining the internal adaptations (as the organization of the features available according to the device type) and the end user is responsible for controlling the adaptation concerning the settings of their profiles and preferences.

[15] This implementation has been done within the scope of the EU-funded FP7 Serenoa Project thanks to the efforts of CTIC team: Javier Escolar, Cristina Cachón and Ignacio Marín, for the purpose of demonstration and validation.

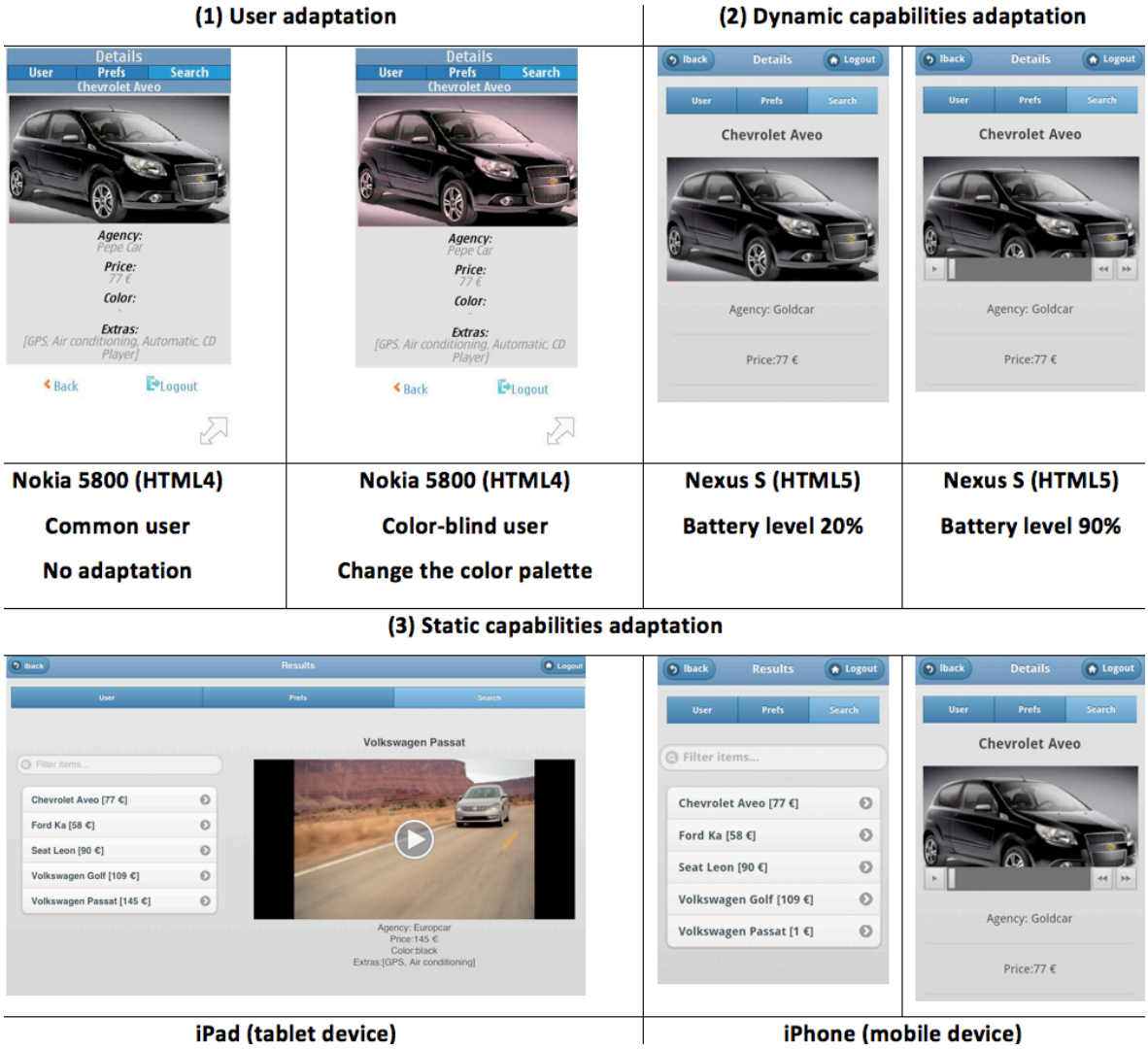


Figure 38. Adapted UI's for car rental according to: (1) the visual impairment of the end user (color-blindness); (2) the battery level (no video is loaded and played); and (3) the static device capabilities, i.e. the device type (an iPad tablet or an iPhone smart phone).

The Figure 39 and Figure 40 present how this third example can be located within the diagrams of the CARF and the CADS. The instantiations illustrated by CADS and CARF cover the three scenarios of example (user-oriented adaptation, dynamic and static information of the contexts). The adapted UI's consider respectively: the color-blindness of the user, the battery level and the static capabilities of the device (tablet PC or smart phone).

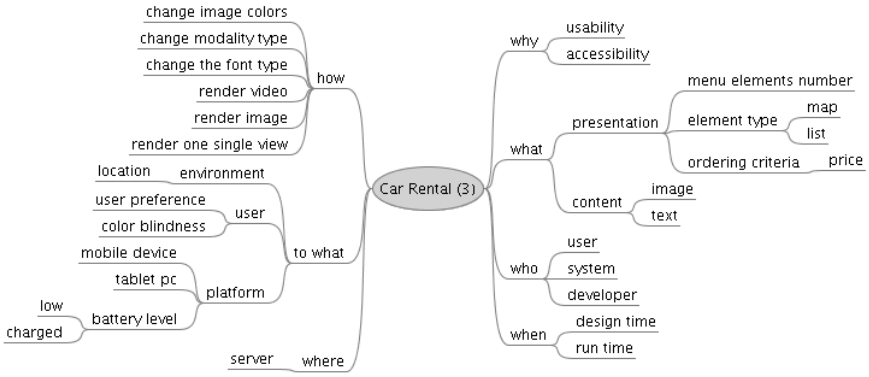


Figure 39. CARF Instantiation of the third implementation of the car rental.

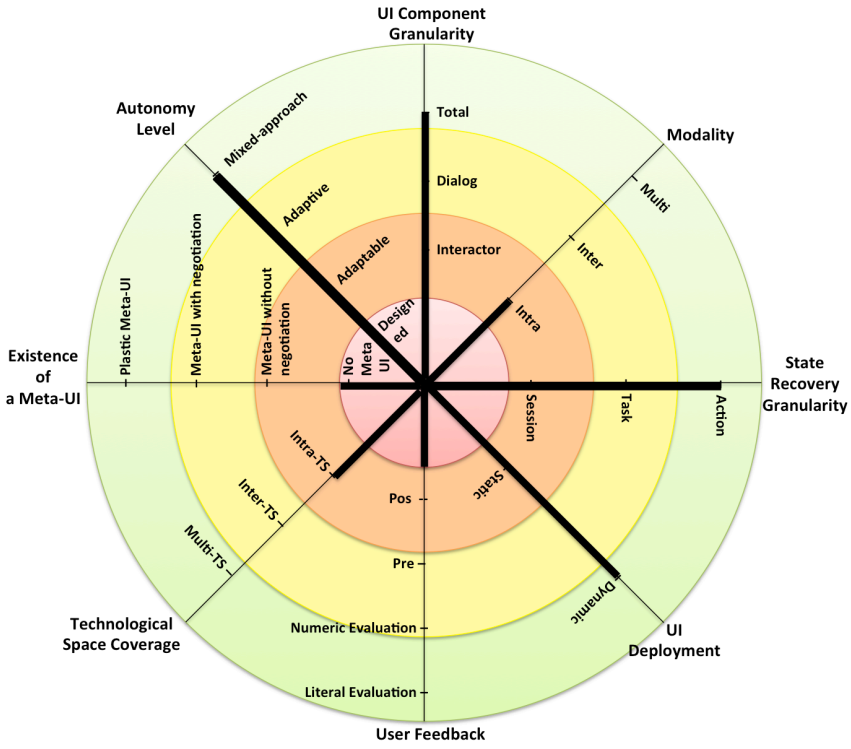


Figure 40. Instantiation of the CADS for the third car rental example.

4.5 Specification of the Touristic Application Case Study

The touristic application[16] consists in a scenario in which the interactive system supports users in the task of retrieving information about a trip. In this sense, three implementations were defined: Walkware, Weather, and Weathaware. Although these applications belong to the specific application domain of tourism, they cover multidimensional aspects of CAA: they comprise different content types (like images, texts and maps), they are generated based on varied context information (like device type and user preferences), and

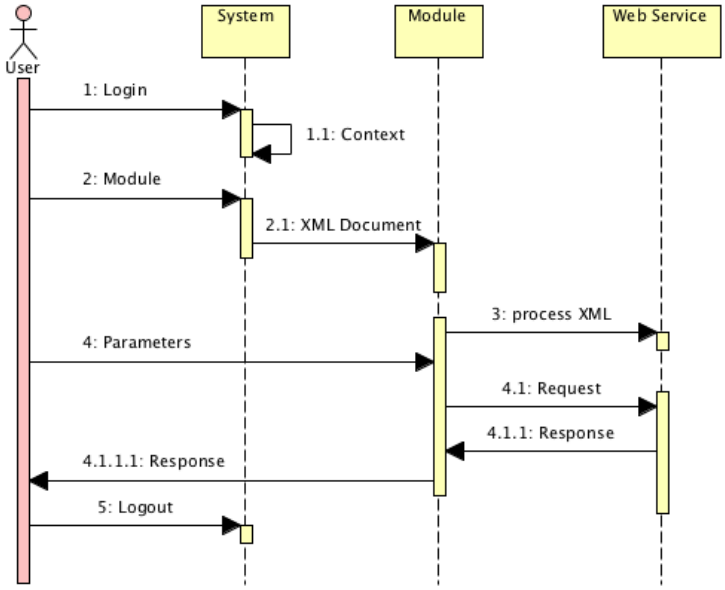
[16] This case study scenario and following implementations have been done thanks to the support of the students: Quentin Poncelet and François Debande within the scope of their master thesis [Deb11].

they also cover both static and dynamic contents (thanks to real-time information retrieved from web services).

The CAA process is executed as follows: first, the user logs in, then, the context information document is created according to the actual context of use, and finally, the user selects one module to access. As a response, the system provides to the user a form to define the request parameters. Once the parameters are submitted, the system generates the request and contacts the web service to obtain the necessary information. The data is then, received, treated, processed and the final user interface is generated, according to: the context of use, the chosen parameters, and the response of the web service. The interaction is concluded when the user logs out. The sequence diagram depicted in Figure 41 illustrates such steps.

Concerning the design decisions: the system was implemented with PHP, the users data, as account information, are stored and managed with a MySQL database, the graphic layouts are defined in CSS style sheets, and the dynamic context information, which is retrieved as soon as the user logs in, are stored in an XML-based document (an excerpt is presented in Figure 42.

Figure 41. Sequence diagram of the touristic application with the sequence of messages exchanged between: users, system, modules, and web services.



The parameters to get the Weather Forecast, for instance, include the country and the amount of days of interest. The request is firstly dynamically generated by the module (according to all the parameters previously specified), and secondly submitted to the Web Service, that replies with the appropriate response. With the resulting contents provided by the Web Service, the Module dynamically generates the final UI contents that are formatted according to the pre-defined style sheets and finally presented to the end user. The interaction finishes when the user logs out.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<Data>
  <Context>
    <Platform>PC</Platform>
    <DisplayStyle>Graphic</DisplayStyle>
    <InformationLevel>Big</InformationLevel>
    <Orientation>Line</Orientation>
    <BatteryLevel>Good</BatteryLevel>
    <Background>Bright</Background>
    <Colors>Bright</Colors>
  </Context>
  <Hardware>
    <Display>
      <height>900</height>
      <width>1440</width>
    </Display>
    <Battery>
      <batteryBeingCharged>false</batteryBeingCharged>
      <batteryLevel>60</batteryLevel>
    </Battery>
  </Hardware>
</Data>
```

Figure 42. An illustrative example of the XML-based document to describe context information, concerning the platform type, orientation of the device, level of battery, etc.

Figure 41 illustrates an overview of the main entities and the sequence and type of messages that they exchange.

4.6 Walkware

The Walkware provides users with a list of cities, surroundings distance, interest points (e.g. culture, markets, hiking), housing (hotel, camp, B&B), and restaurants. As a response, users obtain a map with icons indicating the location of the parameters of interest. Users can also customize and refine their request afterwards by selecting for instance the start point, the end point and the specific means of transport (driving or walking).

In Walkaware, by default the surroundings distance for the map is of 20km for smart phones, 50km for tablet PC's, and 100km for desktop PC's. This module uses Google Maps as a web service. By default maps are rendered: in small dimension for smart phones, medium dimensions for tablet PC's and large dimensions for desktop PC's.

Figure 43 presents six adaptation rules that have been implemented for the Walkaware module. The rules were structured in conditions and actions. The conditions, presented in the top part of the table, concern the device type (smart phone, tablet PC or desktop PC) and also the end user location (USA, Liberia and Birmania). The actions, presented in the bottom part of the table, concern the size of the text, icons and maps (small, medium and large), the search distance (20km, 50km or 100km) and also the unit system (imperial or metric) according respectively to the screen size and the user location.

The result of the application of such rules can be visualized in the Figure 44, Figure 45, Figure 46, Figure 47, Figure 48, and Figure 49. They illustrate the interfaces for selection of parameters of interest (for the request) and also the resulting interfaces. They cover the 3 different platforms: smart phones, tablet PC's and desktop PC's.

Conditions	01	02	03	04	05	06
Platform = Smartphone	T	F	F	T	F	F
Platform = Tablet	F	T	F	F	T	F
Platform = Desktop	F	F	T	F	F	T
Location = USA/Liberia/Birmania	T	T	T	F	F	F
Actions						
Text size = Large	X	X		X	X	
Text size = Small			X			X
Icon size = Small	X			X		
Icon size = Medium		X			X	
Icon size = Large			X			X
Google maps size = Small	X			X		
Google maps size = Medium		X			X	
Google maps size = Large			X			X
Search distance = 20 km	X			X		
Search distance = 50 km		X			X	
Search distance = 100 km			X			X
Imperial system	X	X	X			
Metric system				X	X	X

Figure 43. Decision table with a sample of 6 adaptation rules for Walkware that has been implemented. Each of the 6 columns on the right part is a rule, with the conditions on top (4 parameters) and actions on the bottom part (13 actions).

The background color is also adapted according to the level of charge of the battery, so for instance if the tablet is running out of battery a dark color (black) is selected for the UI, saving the charge.

Although the users have a limited region of the map rendered according to their space available in the screen, they are able to change the zoom of the image, according to their needs.

The adaptation rules have been extracted from the Cards provided in CARF. The implementation of the applications has been supported by the definitions of CAMM, and then analyzed with CADS.

Although not all the dimensions of the context and all the system aspects have been covered, TriPlet proved to be a useful source of information to both define and implement CAA in this scenario.

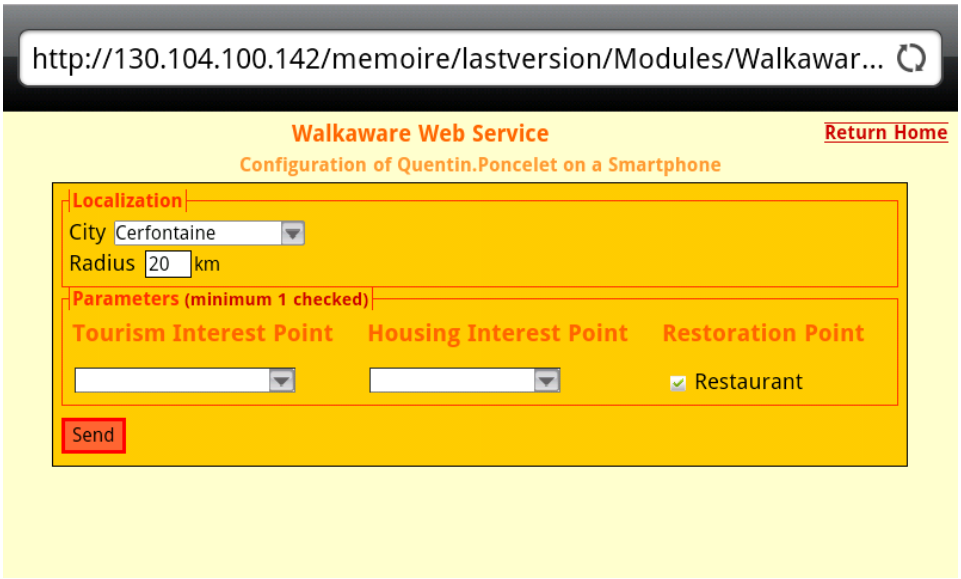


Figure 44. Walkaware for smart phone: request form in a normal battery level.

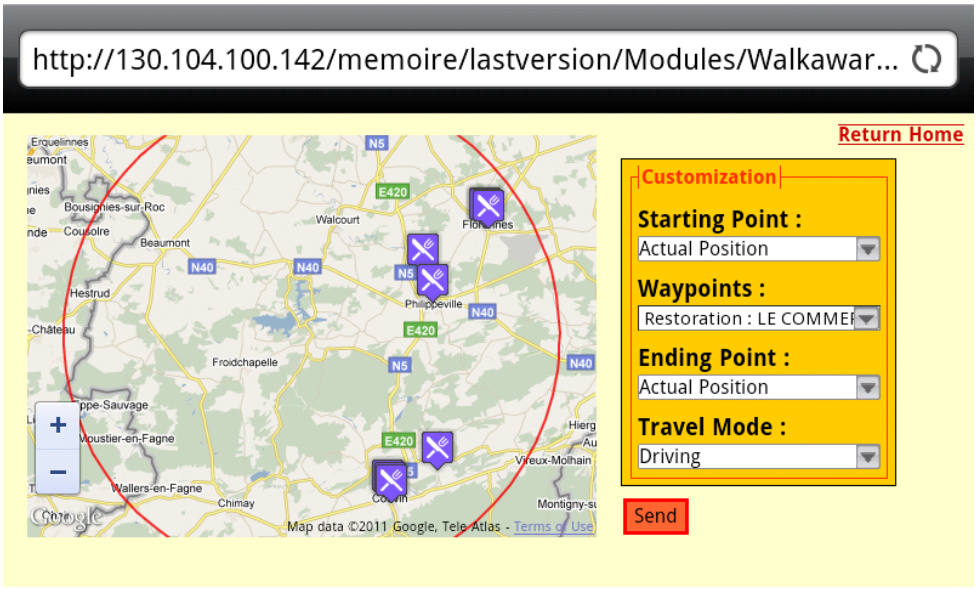


Figure 45. Walkaware for smart phone: results of the request in a normal battery level.

This example covers a graphical modality of interaction, and its UI elements are also suitable for touch-based interaction.

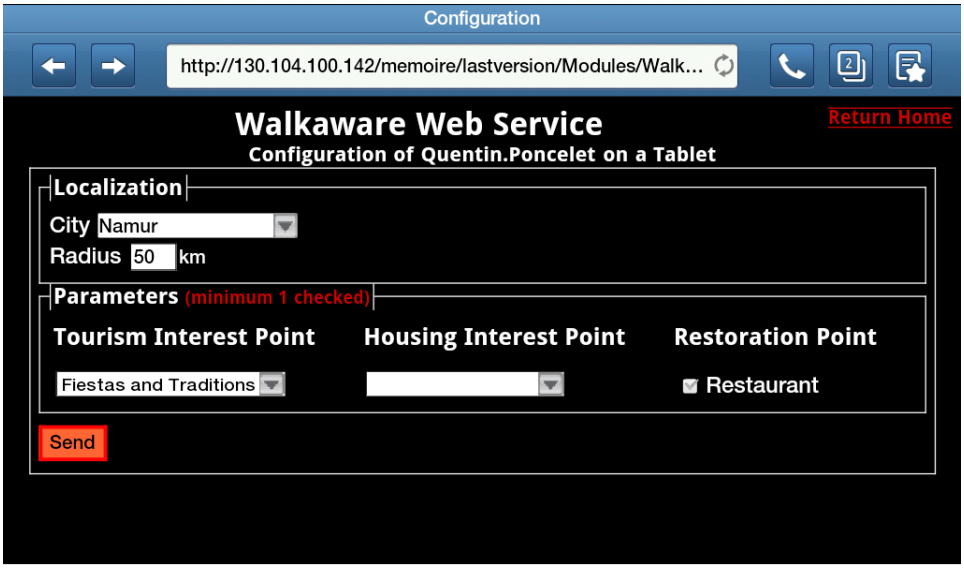


Figure 46. Walkaware for tablet PC: request form in a low battery level.

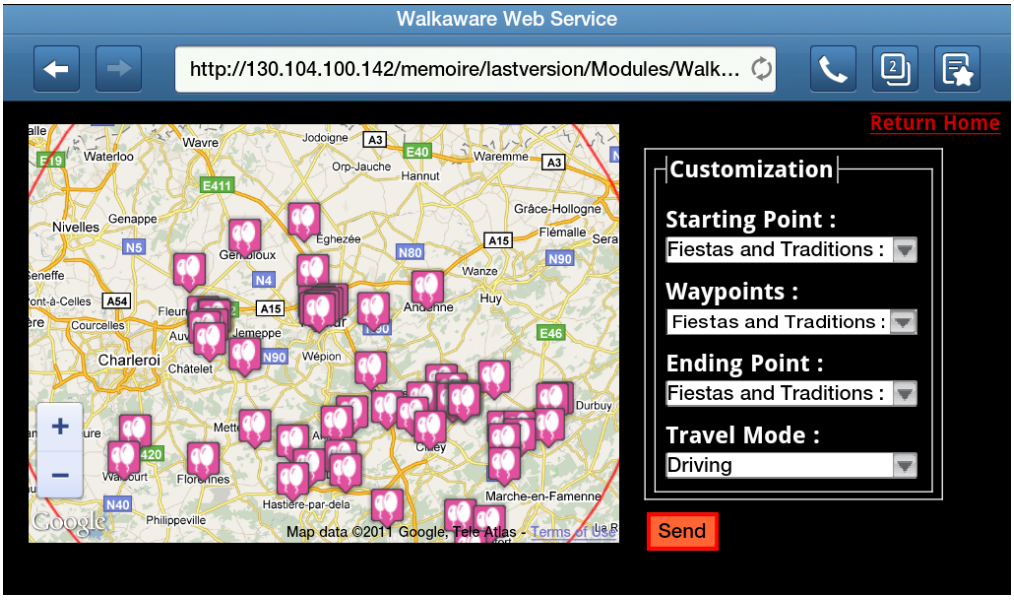


Figure 47. Walkaware for tablet PC: results of the request in a low battery level.

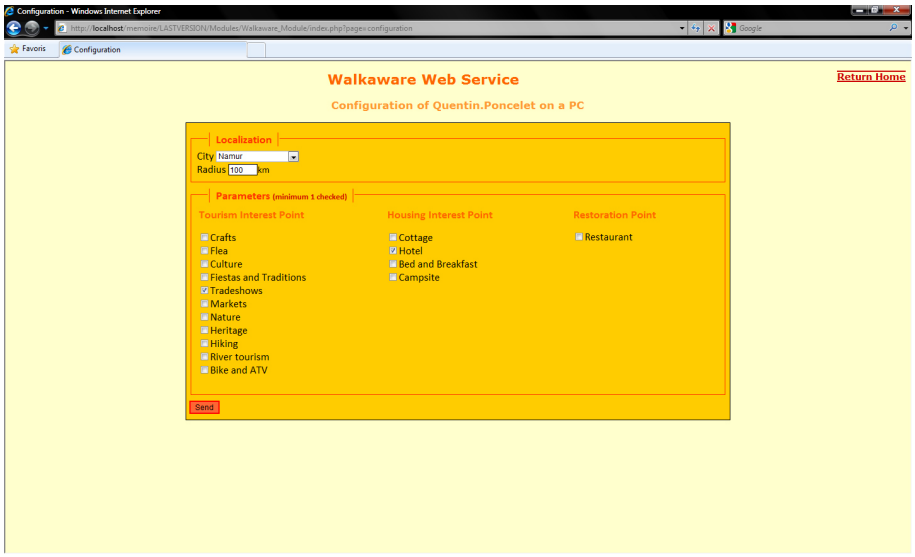


Figure 48. Walkaware for desktop PC: request form.

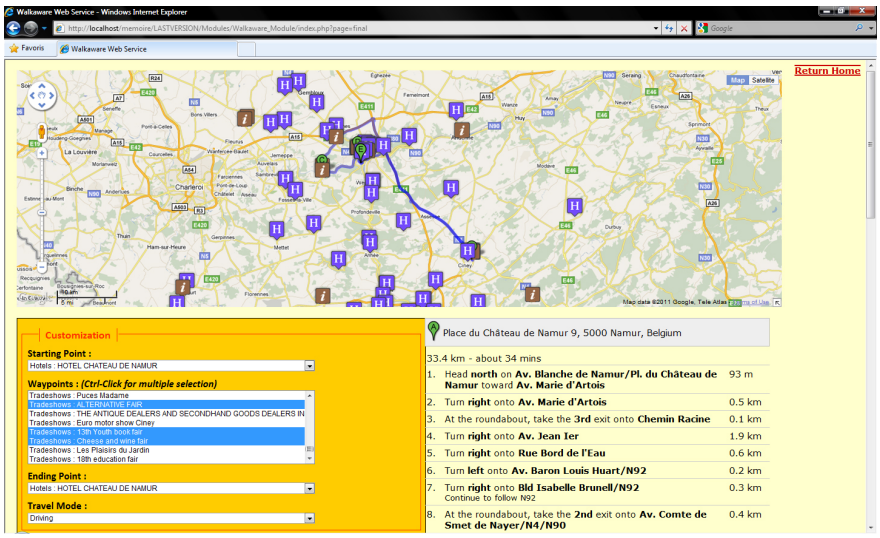


Figure 49. Walkaware Forecast UI adapted for a desktop PC.

The instantiation of the conditions of the CAA rules vary according to the size of the device of the end user, i.e. the smart phone, the tablet PC and the desktop PC.

The contexts covered by this second example combine information from users, environments and platforms, and the systems aspects that are subject to the adaptation involve the contents, the presentation and the navigation. The classes of CAMM are instantiated in these examples in the following manner:

- As context, the preferences of the users are considered, and also the device type concerning the platforms, and the location of the environment are covered, more precisely users preferences regarding the parameters selected in the request form, the type of the device and its level of charge of the battery, and the environment location are used as context information.

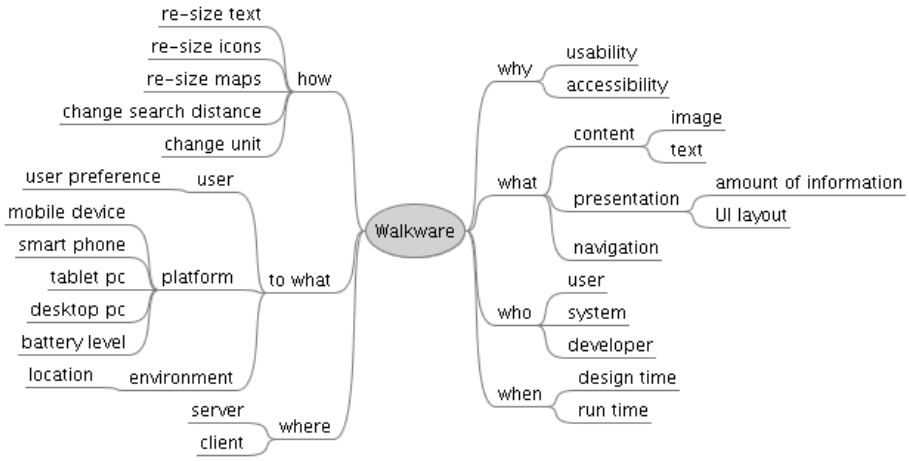


Figure 50. Instantiation of the CARF of the touristic application Walkware.

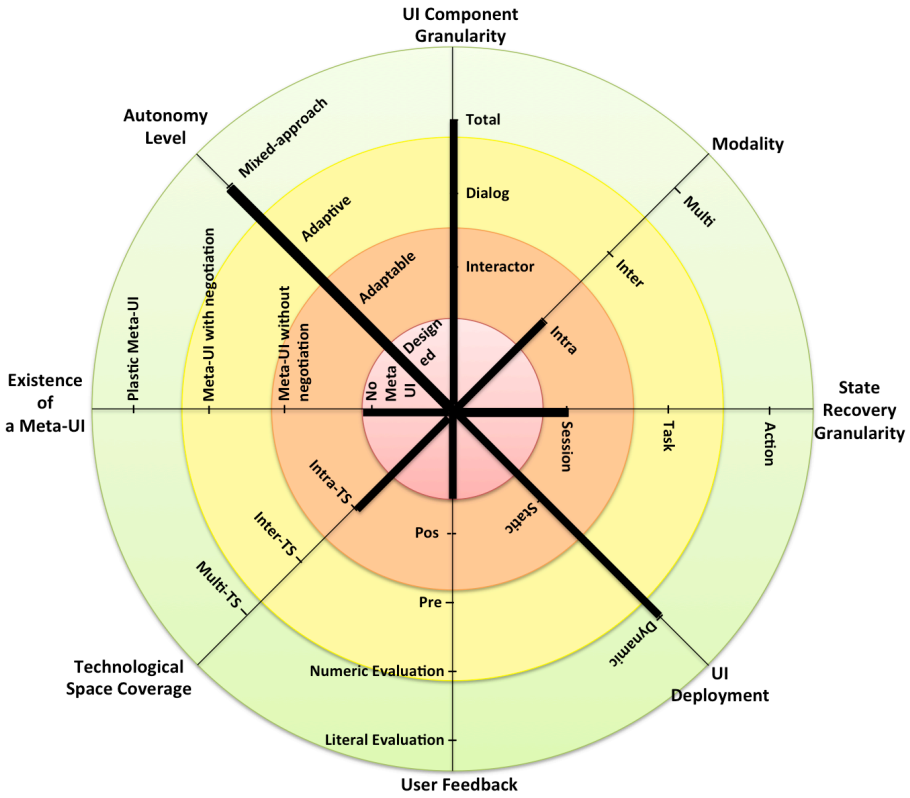


Figure 51. Instantiation of the CADS of the touristic application Walkaware.

♦ The adaptation rules cover the contexts of use described above and change the contents of the UI (e.g. the information available in the UI, and the metric units), the presentations of the UI (e.g. layout of the UI units and colour of the background), and the navigation (the structure of the tasks and the amount of features per UI may change to better suit the different scenarios of use).

♦ As models mainly the task tree (reflected in the definition of the UI and their arrangements) and the final user interfaces have been affected.

♦ As agent the developer is responsible for defining the internal adaptation (e.g. the organization of the features available per UI) and the end user is responsible for controlling the adaptation concerning the different information presented in the UI.

The Figure 50 and Figure 51 present how this first example of the touristic application is located within the CARF and the CADS. These instantiations cover the three scenarios of example for different platforms.

4.7 Weather

The Weather enables users to verify the weather forecast based on a set of request parameters, such as: the amount of days the users want to access in the weather forecast results (1 to 6), the location of interest (Atlanta, Boston or Miami), the display modality (graphic or text), specific information (temperature, weather icons, precipitation, wind), and the chart type (cloud, precipitation and temperature).

According to the screen size of the platform (device) in use, the amount of parameters to set the request changes, as well as the amount of results that are presented to the end user and the modality type. Table 7 presents for the Weather Module, the adaptation rules (platform conditions vs. actions) defined concerning the amount of information presented by default in the UI (parameters and results) and the modality type.

Table 7. The adaptation rules defined are for the Weather Module, concerning the amount of information presented by default in the UI.

Actions	Smart phone	Tablet PC	Desktop PC
# of Parameters	3	5	9
# of Results	2	4	6
Modality Type	text	graphic	graphic

Figure 52 illustrates the example for the desktop PC. Note that users are able to manually select any of the parameters available for their request. Furthermore, the user preferences always have higher priorities than the system decisions.

The adaptation for the Weather module also considers the level of battery charge, in this sense, if there is less than 20% of charge the state is considered as low, otherwise it is considered as good. For low levels of battery, the colors of the UI background are dark (black) and for good levels of charge, they are bright (blue).

The same rule applies according to the period of the day, in this sense: during the day (higher brightness levels in the environment) the background is bright (blue) and the content is dark, and during the night (lower light levels), the background is dark (black) and the content is bright.

The level of light in an environment is strongly influenced by other factors as well. Light sensors, as available in smart phones, are able to achieve more precise results to get this specific context information. Besides this, to perform a more efficient adaptation of the colors and contrast levels in the GUI's, other factors such as the task of the user, his or her preferences, visual impairments, and geographic location must be equally taken into account and prioritized accordingly. In our approach, it is enough to extend the list of conditions in the decision table to cover also further scenarios; this can be done either manually by the user or automated by the system.

Figure 52. Weather for desktop PC: User form for defining the request parameters.

CONDITIONS	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18
Platform = Smartphone	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F
Platform = Tablet	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F
Platform = Desktop	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T
Location = Seashore	T	T	T	F	F	F	F	F	F	T	T	T	F	F	F	F	F	F
Location = Plain	F	F	F	T	T	T	F	F	F	F	F	F	T	T	T	F	F	F
Location = Mountain	F	F	F	F	F	F	T	T	T	F	F	F	F	F	F	T	T	T
Location = USA/Liberia/Birmania	T	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F	F
ACTIONS																		
Forecast days = 2	X			X			X			X			X			X		
Forecast days = 4		X			X			X			X			X			X	
Forecast days = 6			X			X			X			X			X			X
Selected parameters = 3	X			X			X			X			X			X		
Selected parameters = 5		X			X			X			X			X			X	
Selected parameters = 9			X			X			X			X			X			X
Text size = Large	X	X		X	X		X	X		X	X		X	X		X	X	
Text size = Small			X			X			X			X			X			X
Icon size = Small	X			X			X			X			X			X		
Icon size = Medium		X			X			X			X			X			X	
Icon size = Large			X			X			X			X			X			X
Parameters + Wind	X	X	X	X	X	X				X	X	X	X	X	X			
Parameters + Wave size	X	X	X							X	X	X						
Parameters + Precipitation level				X	X	X							X	X	X			
Parameters + Snow level							X	X	X							X	X	X
Imperial system	X	X	X	X	X	X	X	X	X									
Metric system										X	X	X	X	X	X	X	X	X

Figure 53. This decision table represents a sample of 18 adaptation rules for weather that has been implemented. Each of the 18 columns on the right part is a rule, with the conditions on top (7 parameters) and actions on the bottom part (17 actions).

Logically larger tables will imply in more processing efforts, and consequently in lower responsiveness rates of the system (depending on processing capabilities available). However such trade-off can be also resolved based on the user preferences.

Figure 53 represents the decision table used to define the adaptation rules for the Weather module. In the top part of the table the 18 possible combinations of the 7 conditions have been defined. Three platform types are envisaged: a smart phone, a tablet PC and a desktop PC. Three types of location of interest have been selected: sea, plain or mountain, as well as three specific locations (USA, Liberia and Burma). These combinations lead to 17 actions (adaptation techniques) that modify the resulting UI.

The UI illustrated in the Figure 54, Figure 55, Figure 56,, Figure 57, Figure 58, Figure 59, and Figure 60 show the examples of adaptation for the smart phone, tablet PC and desktop PC. Not only the amount of information changes according to the space available in the device screen (2, 4, or 6 days for the weather forecast), but also: the amount of parameters for selection (3, 5, and 9), the text and icons size (small or large), the additional information provided (wind, waves, precipitation, and snow), and the unit system (imperial or metric).

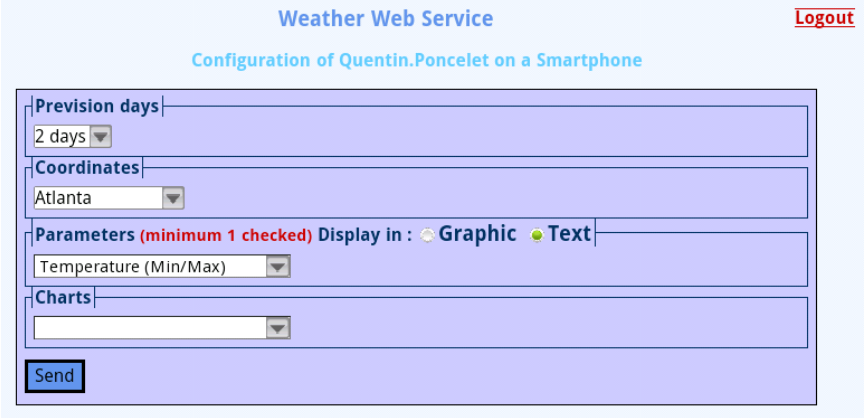


Figure 54. Weather for smart phone: request form in a normal battery level.

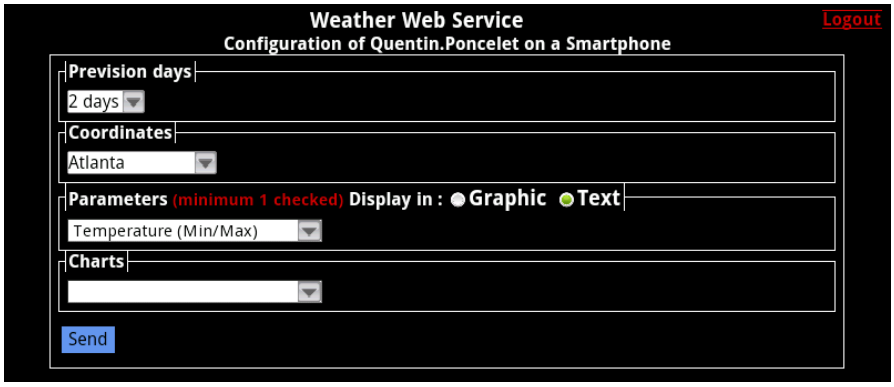


Figure 55. Weather for smart phone: request form in a low battery level.

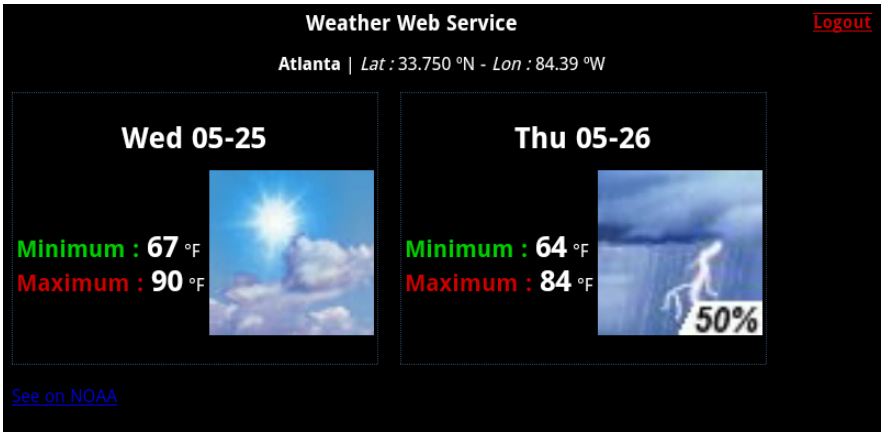


Figure 56. Weather for smart phone: results of the request in a low battery level.

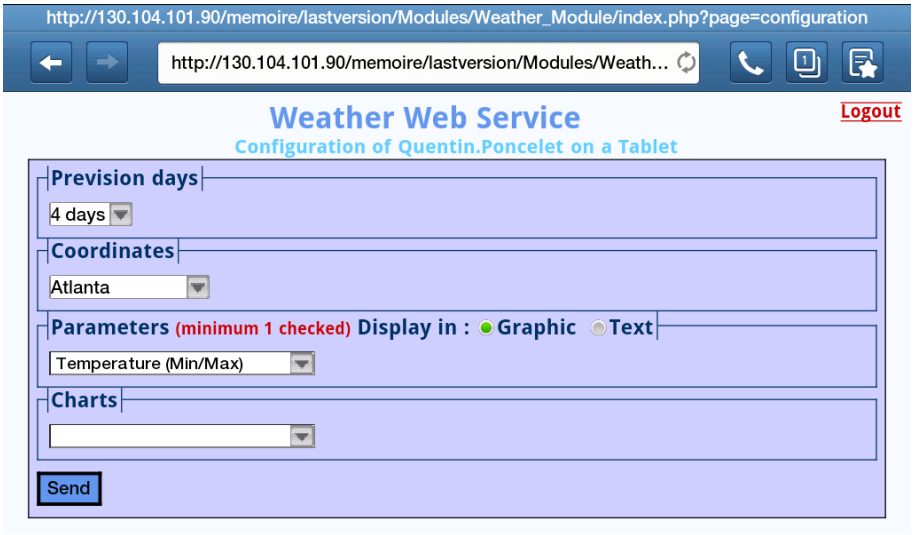


Figure 57. Weather for tablet PC: request form.

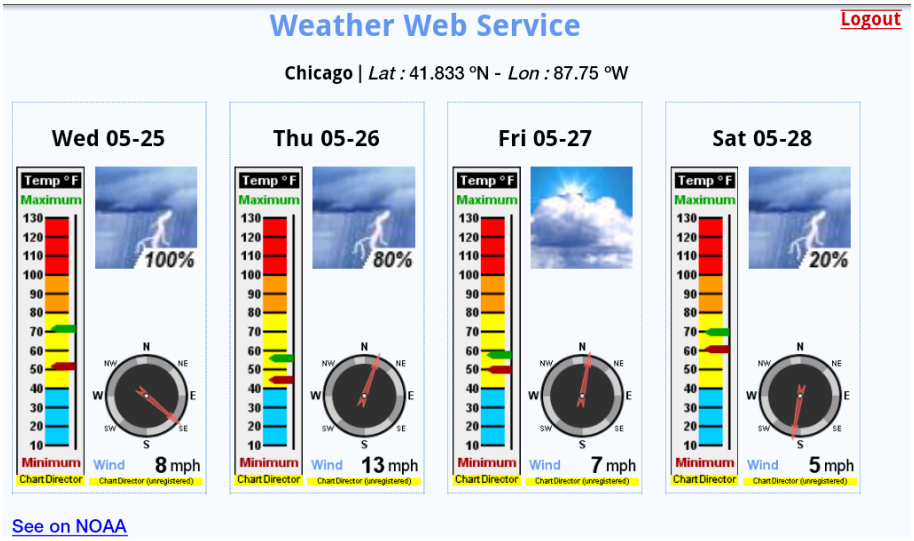


Figure 58. Weather for tablet PC: results of the user request.

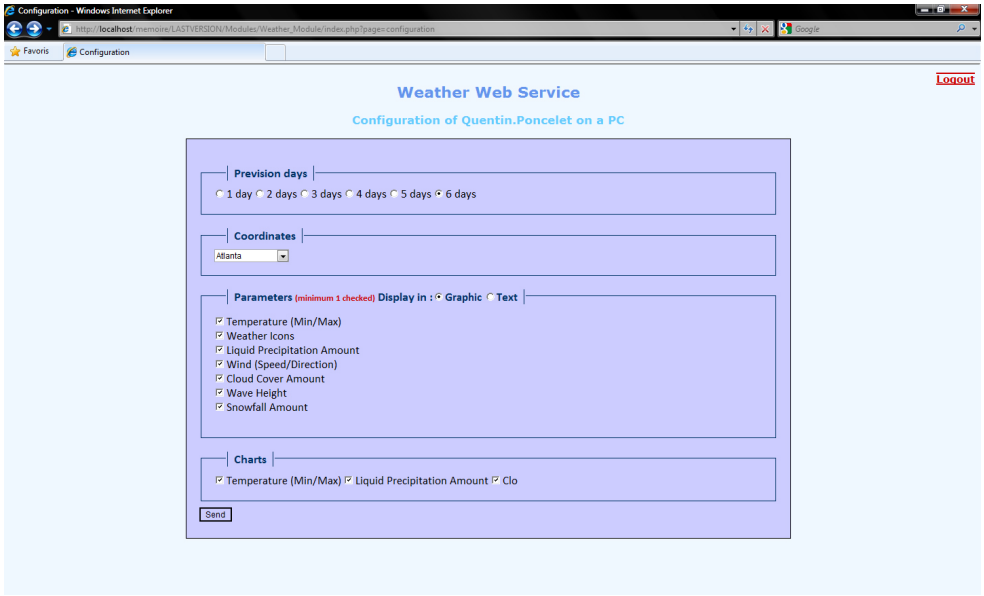


Figure 59. Weather for desktop PC: request form.

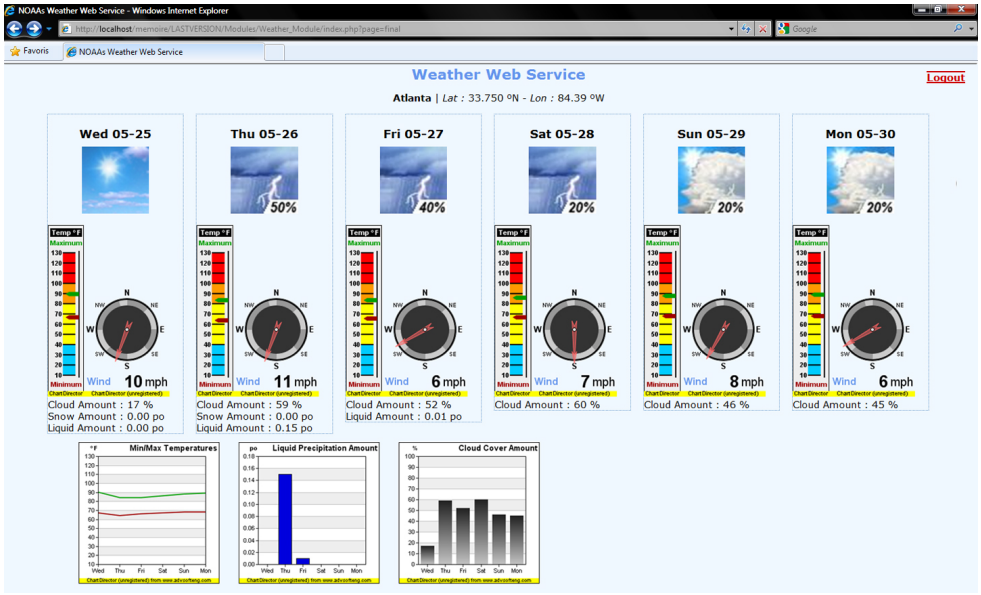


Figure 60. Weather Forecast UI adapted for a desktop PC: the forecast covers 6 days, and several parameters (amount of clouds, snow, liquid, temperature, precipitation, etc.).

The instantiation of the conditions of the CAA rules vary according to the size of the device of the end user, i.e. the smart phone, the tablet PC and the desktop PC, and also to the location of the end user.

The contexts covered by this second example (Weather) combine information from users, environments and platforms, and the systems aspects that are subject to the adaptation involve the contents, the presentation and the navigation. The classes of CAMM are instantiated in these examples in the following manner:

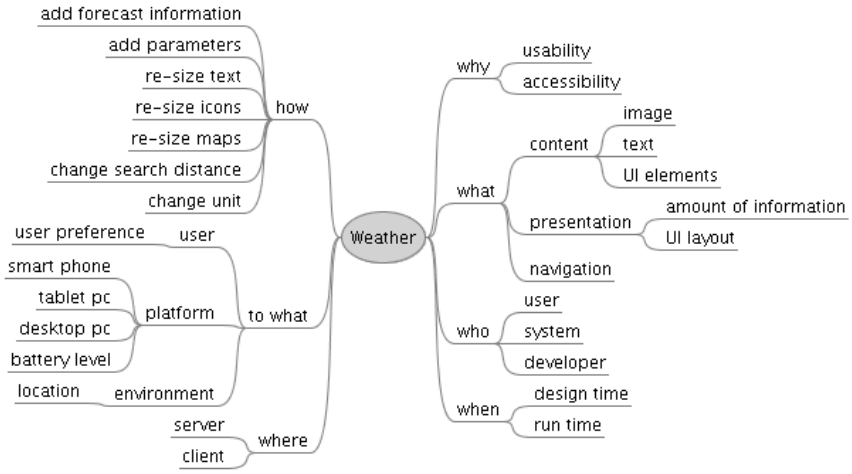


Figure 61. Instantiation of the CARF of the touristic application Weather.

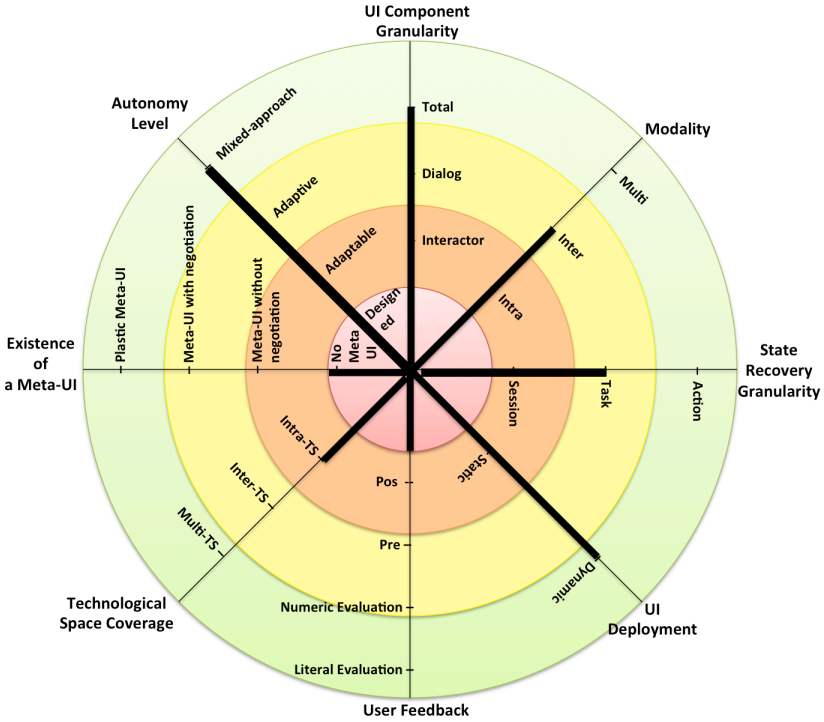


Figure 62. Instantiation of the CADS of the touristic application Weather.

♦ As context, the preferences of the users are considered, the platforms, and the location of the environment are covered, more precisely users preferences regarding the parameters selected in the request form, the type of the device and its level of charge of the battery, and the environment location and light levels are used as context information.

♦ The adaptation rules cover the contexts of use described above and use them to appropriately change the contents of the UI (e.g. the amount, and the type of the information available in the UI, as well as the metric units for the temperature), the presentations of the

UI (e.g. layout of the UI units and colour of the background), and the navigation (the structure of the tasks and the amount of features per UI may change to better suit the different scenarios of use).

- As models mainly the task tree (reflected in the definition of the UI and their arrangements) and the final user interfaces have been affected.
- As agent the developer is responsible for defining the internal adaptation (e.g. the organization of the features available per UI) and the end user is responsible for controlling the adaptation concerning the different information presented in the UI.

The Figure 61 and the Figure 62 present how this second example of the touristic application (Weather) can be located within the CARF and the CADS.

The instantiations illustrated by the figures cover the three scenarios of example for different platforms.

4.8 Weathaware

The Weathaware combines Weather and Walkware, permitting users to select: the days for the trip, the location, points of interest (attractions, accommodation and restaurants), the presentation modality (graphic or text) and also parameters of interest (temperature, weather, wind and clouds). In return, users obtain the map with icons locating the information of interest on it (according to their previous selection), the information of interest, and the option to customize the request, i.e. to refine the results by selecting the starting point, the intermediary points, the end point, and the transport means.

The composition module (Weathaware) respects the same default settings as their source modules. Figure 63, Figure 64, Figure 65, Figure 66, Figure 67, and Figure 68 illustrates the adapted interfaces of the Weathware the figures respectively refer to the request form and the resulting UI for the three platforms considered: smart phone, tablet PC and desktop PC.

The request forms enables users to select their preferences and information of interest, as the amount of days, locations, points of interest, etc. The resulting adapted UI, includes the maps (with points of interest), the location information (route) and the weather information. The results can be refined, by selecting also the destination, stops and travel modes.

Figure 63, Figure 64, Figure 65, Figure 66, Figure 67, and Figure 68 illustrates the variation in the amount, type of contents and options available for end users to select and access. As such, for a desktop PC users have a larger map image displayed, covering a larger region, several textual descriptions are exhibited providing the directions of interest, and also additional fields are presented to refine the results obtained. For the smart phone context, fewer fields are available for the search, and the results are more limited too (e.g., with a smaller map image).

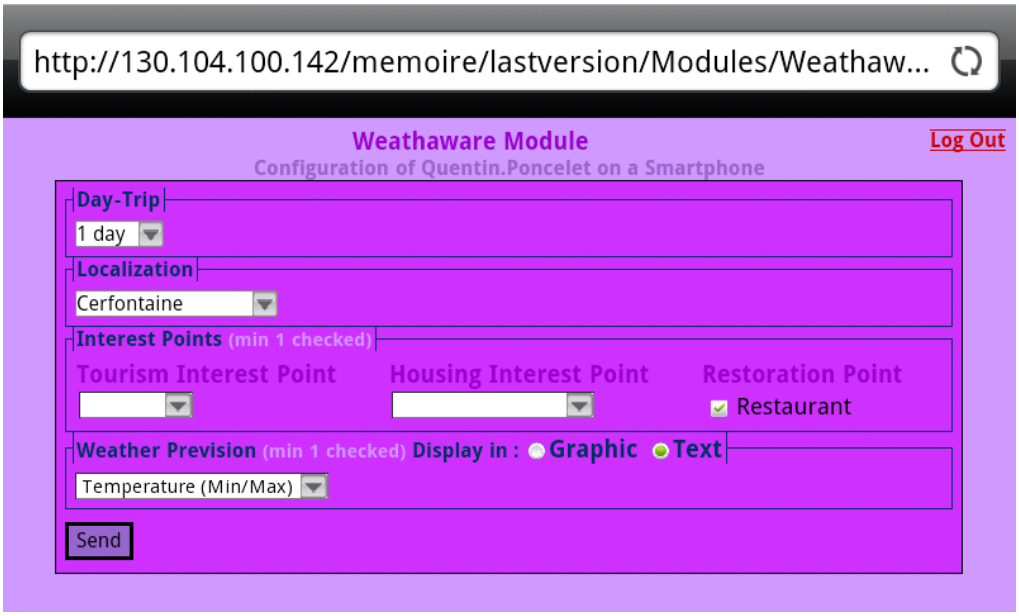


Figure 63. Weathaware for smart phone: request form.

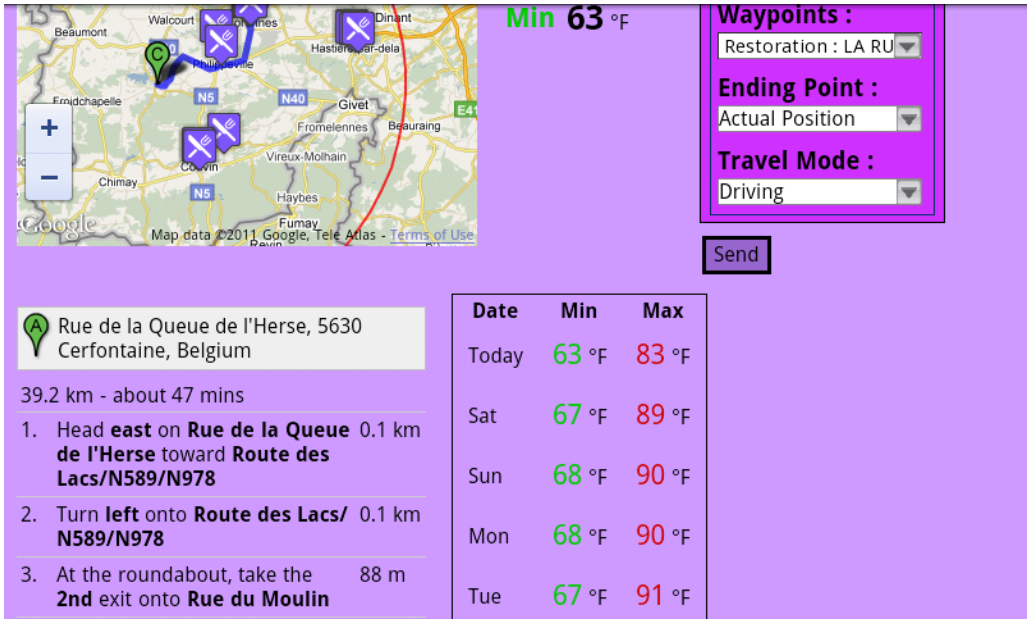


Figure 64. Weathaware for smart phone: results of the request.

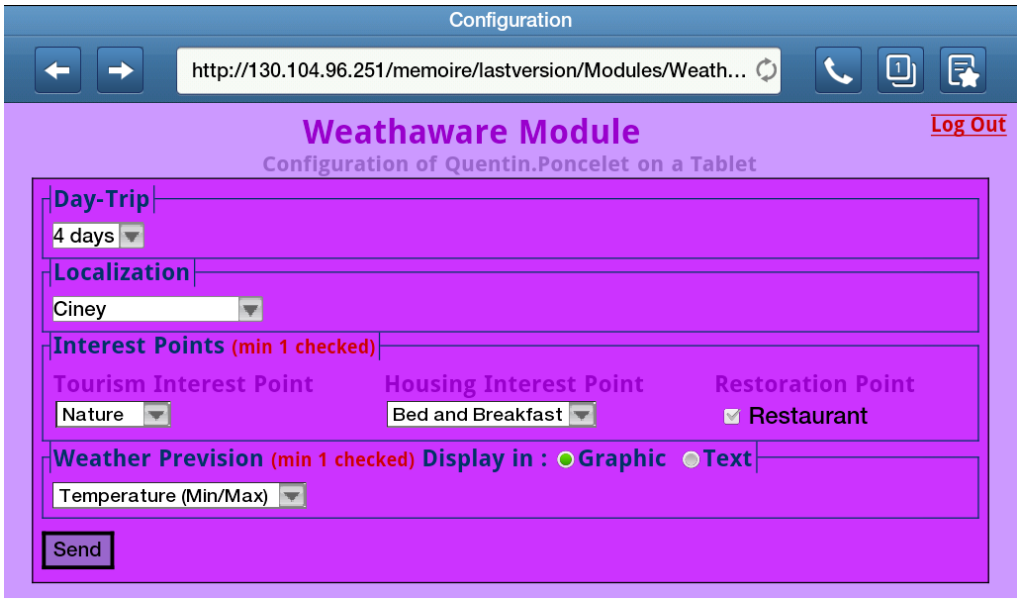


Figure 65. Weathaware for tablet PC: request form.

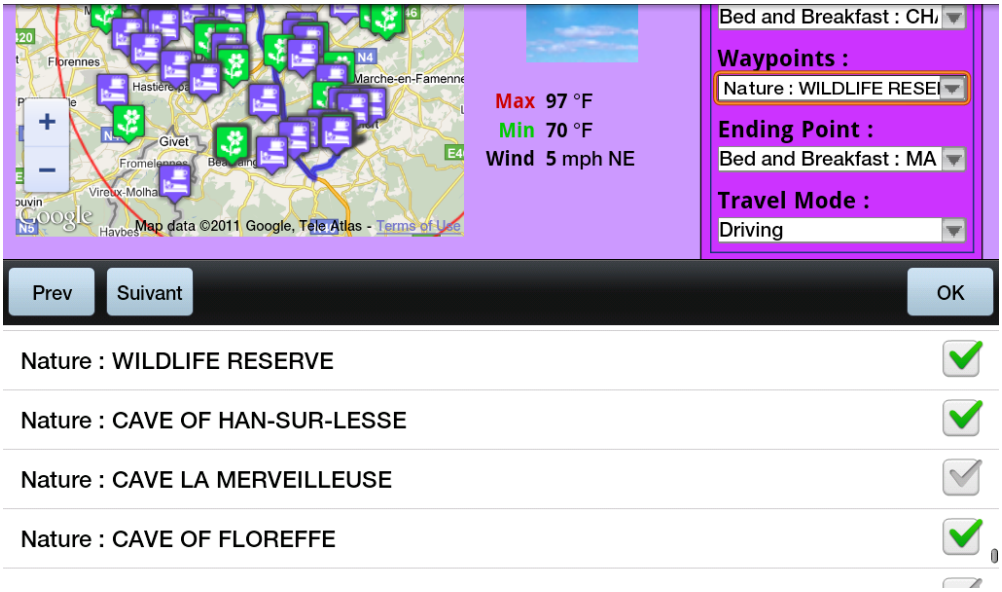


Figure 66. Weathaware for tablet PC: results of the request.

The instantiation of the conditions of the CAA rules vary according to the size of the device of the end user, i.e. the smart phone, the tablet PC and the desktop PC, and also to the preferences of the end user (according to their choice of parameters of interest).

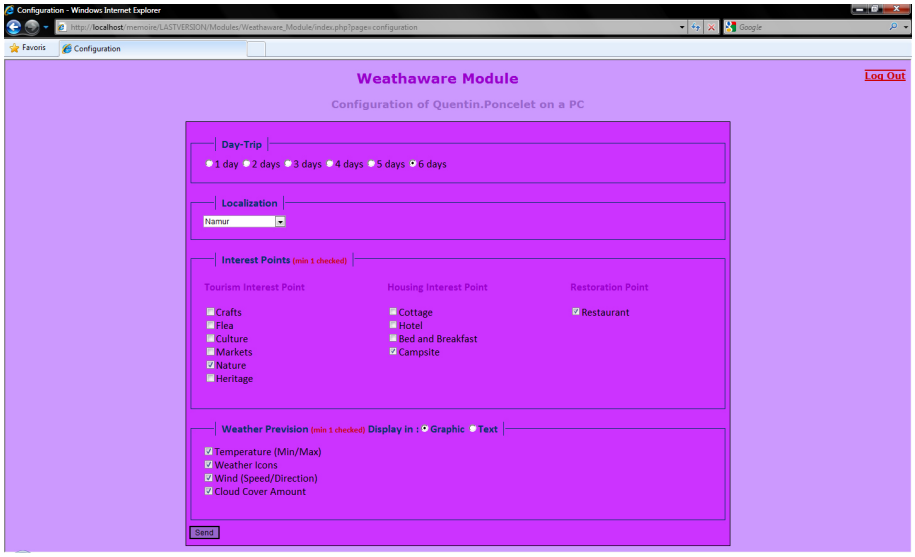


Figure 67. Weathaware for desktop PC: request form.

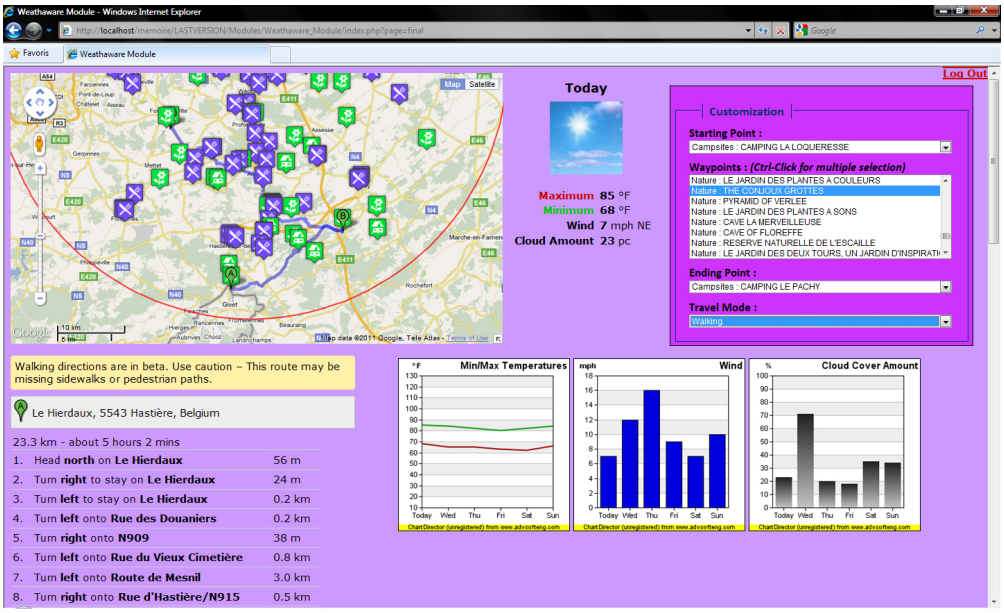


Figure 68. Weathaware for desktop PC: results of the request.

The contexts covered by this second example (Weathaware) combine information from users, environments and platforms, and the systems aspects that are subject to the adaptation involve the contents, the presentation and the navigation. The classes of CAMM are instantiated in these examples in the following manner:

- As context, the preferences of the users are considered, the platforms, and the location of the environment are covered, more precisely users preferences regarding the parameters selected in the request form, the type of the device and its level of charge of the battery, and the environment location are used as context information.

♦ The adaptation rules cover the contexts of use described above and use them to appropriately change the contents of the UI (e.g. the amount, and the type of the information available in the UI, as well as the metric units for the temperature), the presentations of the UI (e.g. layout of the UI units and colour of the background), and the navigation (the structure of the tasks and the amount of features per UI may change to better suit the different scenarios of use).

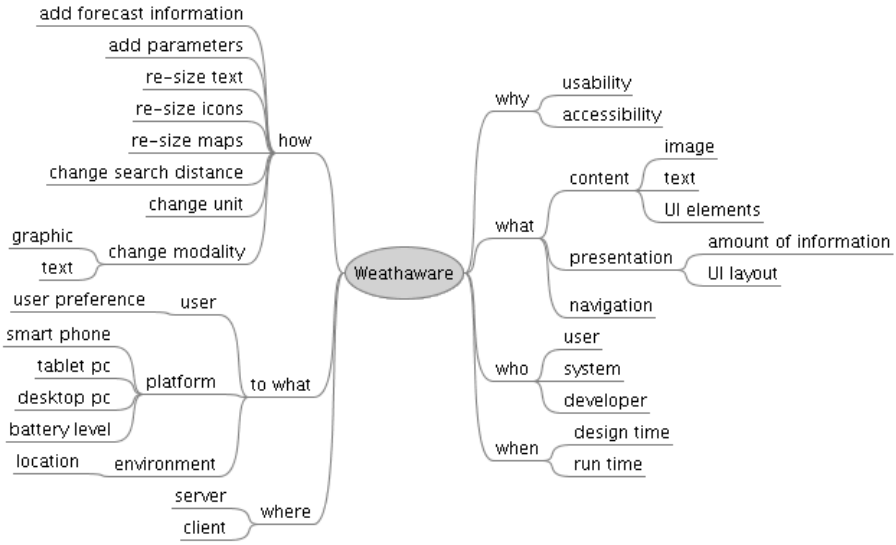


Figure 69. Instantiation of the CARF of the touristic application Weathaware.

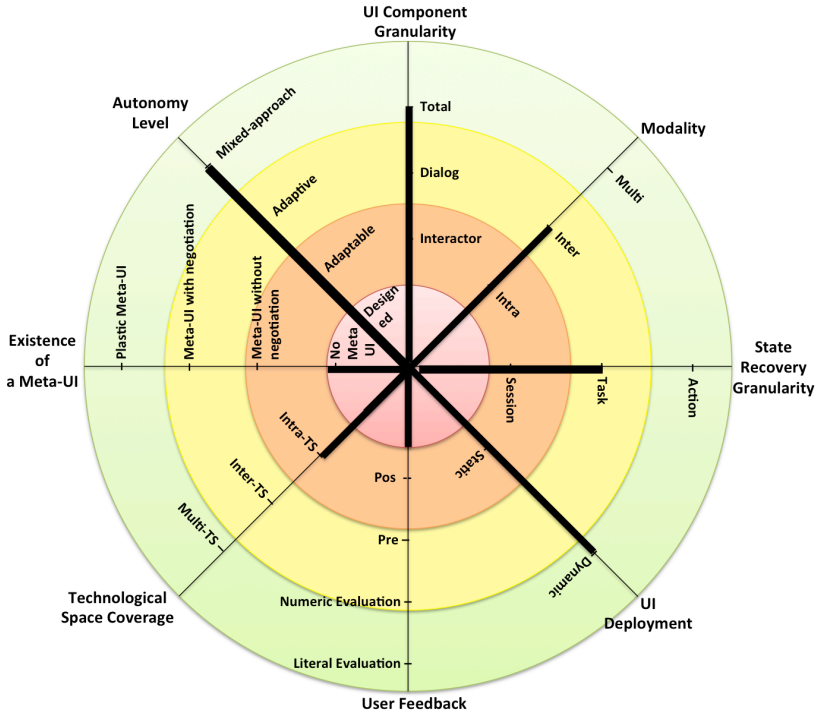


Figure 70. Instantiation of the CADS of the touristic application Weathaware.

♦ As models mainly the task tree (reflected in the definition of the UI and their arrangements) and the final user interfaces have been affected.

- As agent the developer is responsible for defining the internal adaptation (e.g. the organization of the features available per UI) and the end user is responsible for controlling the adaptation concerning the different information presented in the UI.

The Figures below present how this second example of the touristic application (Weathaware) can be located within the CARF (Figure 69) and the CADS (Figure 70).

The instantiations previously illustrated cover the three scenarios of example for different platforms.

4.9 Discussion

This section presented 2 case study scenarios exemplifying the application of TriPlet in several implementations. For the implementations, the system specifications concerning CAA were defined based on the theoretical components of TriPlet. The implementations of both the car rental and the touristic application, have been defined by means of TriPlet independently of: (i) the contextual information selected (user profile, environment type, and platform used), (ii) the technology chosen for the implementation (programming, scripting and markup languages, architectural approaches) and (iii) the application domain (car rental, tourism). Such support was possible due to the fact that TriPlet covers a conceptual definition in a high level of abstraction.

The selection of the case studies presented in this chapter envisaged covering multiple dimensions of adaptation, for instance by benefiting from different contexts of use, as for example: different users' profile (experienced, color-blind users), different devices and platforms (a mobile phone, a tablet PC, large screens, different battery levels), and also different environments (relaxed vs. stressful, light vs. bright, multiple locations). It is not possible to be exhaustive and to cover all the definitions as TriPlet proposes, however the instantiations are an attempt to cover a significant variety of aspects.

First, CAMM has been used to support the definitions, the design decisions and to structure the CAA's. In the descriptions of the case studies, we highlight the main concepts covered regarding the 4 core definitions of the CAMM: agents, rules, contexts and models employed during the development phases. The case studies covered a variety of contexts (information from user, platform, and environments), multiple agents (user, system, developers), and also a variety of system aspects (content, presentation and navigation). The models employed concern mainly, task and domain, and final user interface.

Then, CARF has been applied to support design decisions to meet the requirements defined for the applications. The target context information has been selected, and the adaptation techniques described by the Cards of the CARF have been employed. For each application implemented, there is a set of corresponding Cards that support the implementation phase. Moreover, there is also an instantiation of the CARF that has been used to define the CAA aspects within the application development phases. These instantiations of CARF are integrated in the descriptions of the case studies.

After the implementations have been concluded, the CADS was applied to locate the instantiations and to analyze the coverage level of CAA of all criteria of CADS. This

enabled us to identify how the CADS dimensions had been explored for the applications. Besides locating the applications within CADS, a comparative analysis was also possible. Figure 71 illustrates the applied CADS. The red, black, and blue axes represent respectively the analyses of the first, the second and the third implementations of the car rental examples.

In this instantiation of the CADS we can clearly notice that regarding *modality*, *user feedback*, *technological space*, and *meta-UI* all the implementations have the same coverage level. On the other hand, for *component granularity*, *state recovery*, and *UI deployment*, the second and third implementations have maximum levels (i.e. total, action, and dynamic), while regarding the *autonomy level* the first implementation is adaptive, the second adaptable, and the third adopts a mixed-approach.

For the implementations of the touristic application the same activities have been performed. Figure 72 illustrates a comparison of the different instantiations for the same criteria of CADS. We notice that the 3 applications: Walkware, Weather and Wethaware cover almost all the same levels of adaptation.

CADS also enabled a comparison between the two case studies. By analyzing the 2 different graphics generated, presented by Figure 71 and Figure 72. The design space proved to be a suitable approach to locate the applications, verify how the dimensions of adaptation have been covered and which areas could deserve further attention in future development efforts in case of extensions, updates or new releases. Moreover CADS has also been proved useful as a means of comparing different instantiations with multiple diagrams.

The case studies aim at exploring the potential of CAA, by benefiting from the guidance, support and solutions provided by TriPlet. To provide an overview of the coverage of TriPlet 3 tables have been created.

The Table 8, Table 9, and Table 10 summarize how the instantiations of the case studies employed the concepts defined by TriPlet. By analyzing such tables it is possible to notice that in general there was a significant variability in the implementations. Specially concerning the contexts and system aspects. For instance, regarding Table 8, the agents, the contexts and the rules have explored different dimensions of CAA, covering the various concepts defined by CAMM. However, concerning the models, we can notice just a partial exploration, i.e. mainly task and final user interface have been employed in the definitions. Given that the implementations do not have a high complexity level, the development phases covered just task and FUI, the model-based approaches could be relevant for extended versions of the Car rental and the Touristic applications.

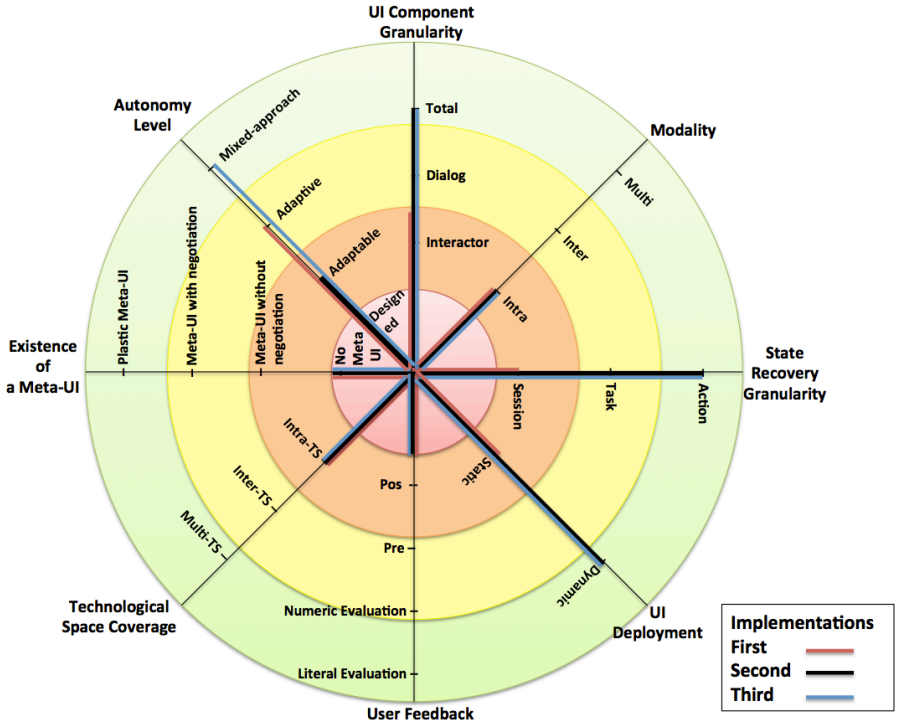


Figure 71. CADS instantiated for the car rental examples: the red axis (first) represents the first implementation version, the black axis (middle) represents the second implementation version and blue axis (last) the third one.

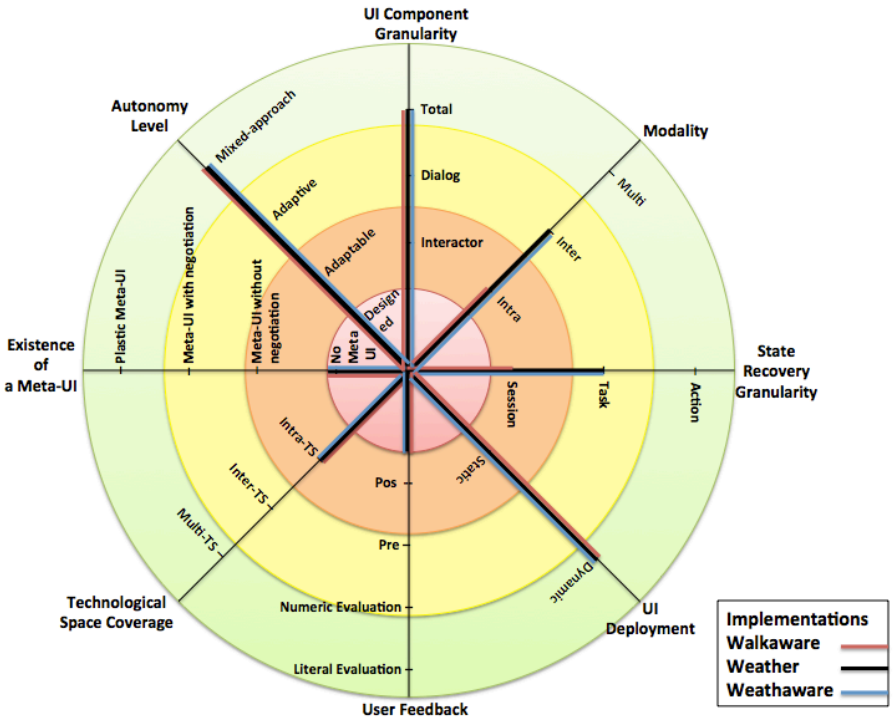


Figure 72. CADS instantiated for the touristic applications the red axis corresponds to Walkware, the black axis corresponds to Weather and blue axis corresponds to Weathaware.

Table 8. CAMM Instantiations for the case studies.

			CAR RENTAL			TOURISTIC APPLICATIONS		
			Example 1	Example 2	Example 3	Walkaware	Weather	Weathaware
CAMM	Agent	User		✔	✔	✔	✔	✔
		System			✔	✔	✔	✔
		Developer	✔	✔	✔	✔	✔	✔
	Context	User	✔	✔	✔	✔	✔	✔
		Platform	✔	✔	✔	✔	✔	✔
		Environment	✔		✔	✔	✔	✔
	Rules	Conditions	User experience, environment	Screen size, browser dimension	Device, color-blindness, battery level	Device type, User location	Device type	Device type, user location
		Actions	Simplify, detail interaction, split tasks	Re-size font, elements, media, change layout	Change colors, modality, font, media	Re-size text, icon, map, change distance, units	Parameters, Results, Modality	Parameters, Results, modality
	Models	Task	✔					
		Final	✔	✔	✔	✔	✔	✔

Table 9. CARF Instantiations for the case studies.

			CAR RENTAL			TOURISTIC APPLICATIONS		
			Example 1	Example 2	Example 3	Walkaware	Weather	Weathaware
CARF	Why	Usability	✔	✔	✔	✔	✔	✔
		Performance	✔					
		Accessibility		✔	✔	✔	✔	✔
	What	Content	✔	✔	✔	✔	✔	✔
		Presentation	✔	✔	✔	✔	✔	✔
		Navigation	✔	✔		✔	✔	✔
	Who	User		✔	✔	✔	✔	✔
		System			✔	✔	✔	✔
		Developer	✔	✔	✔	✔	✔	✔
	When	Design time	✔		✔	✔	✔	✔
		Run time		✔	✔	✔	✔	✔
	Where	Server	✔		✔	✔	✔	✔
		Client		✔		✔	✔	✔
	To what	User	✔	✔	✔	✔	✔	✔
		Platform	✔	✔	✔	✔	✔	✔
		Environment	✔		✔	✔	✔	✔
	How		Simplify UI elements, clarify interaction steps, split tasks, detail interaction	Re-size UI elements, Change layout	Change font, modality, colors, replace media	Re-size text, icons, maps, Change distance, unit, parameters	Re-size text, icons, maps, Change distance, unit, parameters	Re-size text, icons, maps, Change distance, unit, parameters

By analyzing Table 9, we verify that regarding the *why* branch, just 3 qualities have been targeted, namely: usability, performance and accessibility, even if several qualities are enumerated by CARF. On the other hand, for all the other branches of CARF, it is possible to observe that the concepts have been further explored, varying more significantly between implementations, i.e. concerning *what* (system aspects), *who* (agents in control), *when* (development phases), *where* (location), *to what* (contexts of use), and *how* (rules) there was more variation between the instantiations.

Analogously to the Figure 71 and Figure 72, the graphical representations of the CADS, Table 10 shows the instantiations of the CADS for comparison of the two case studies and their respective implementations. We clearly observe a trend in such instantiations, i.e. while for User Interface Component Granularity (UICG), Modality, System Recovery Granularity (SRG), User Interface Deployment (UID), Existence of a Meta-UI and Autonomy a variation can be noted, for the dimensions of User Feedback and Technological Space Coverage (TSC), no variety is observed. The user feedback has not been considered for any of the implementations, even if for some of them (Car Rental 2 and 3) the user had control of the adaptation process to some extent. Concerning the TSC, there is no variation given that all implementations have been defined in the same language, and with settings that do not vary depending on platform.

Table 10. CADS Instantiations for the case studies.

			CAR RENTAL			TOURISTIC APPLICATIONS		
			Example 1	Example 2	Example 3	Walka-ware	Weather	Weatha-ware
CADS	UICG	Interactor	✔					
		Total		✔	✔	✔	✔	✔
	Modality	Intra	✔	✔	✔	✔		
		Inter					✔	✔
	SRG	Action		✔	✔			
		Task					✔	✔
		Session	✔			✔		
	UID	Static	✔					
		Dynamic		✔	✔	✔	✔	✔
	Feed-back	None	✔	✔	✔	✔	✔	✔
	TSC	Intra	✔	✔	✔	✔	✔	✔
	EMUI	None	✔	✔	✔	✔	✔	
		Meta-UI						✔
	Auton-omy	Adaptable		✔				
		Adaptive	✔					
		Mixed			✔	✔	✔	✔

The definitions of these case studies provide an overview about how TriPlet can support CAA, still it is not possible to be exhaustive, in order to cover all contextual information, system aspects, application domains, and technological spaces. In the validation of this thesis, a variety of scenarios for CAA has been chosen, aiming to prove the suitability of the TriPlet by adopting a sample of use cases. Such use case scenarios vary regarding

their design decisions, we can assume that the same support is also feasible regardless of context, system aspects, adaptation rules or technological spaces.

By instantiating TriPlet components, their applicability could be analyzed. CAMM, CARF and CADS proved useful to support stakeholders in the CAA definition and to analyze the adaptation levels.

The validation phase is followed by the analysis phases, i.e. the evaluation of the feasibility and the applicability of TriPlet.

Although the models have not been deeply covered by the instantiations, there has been substantial work on defining CAA for the car rental example across abstraction levels. These efforts have been documented by Schramme [Sch13] and are publicly available online at http://sites.uclouvain.be/mbui/. This web page provides a model voyager, a platform that stores several instantiations of the car rental example and also their respective models. A dynamic tree enables end users to navigate through the models and to visualize their different representations.

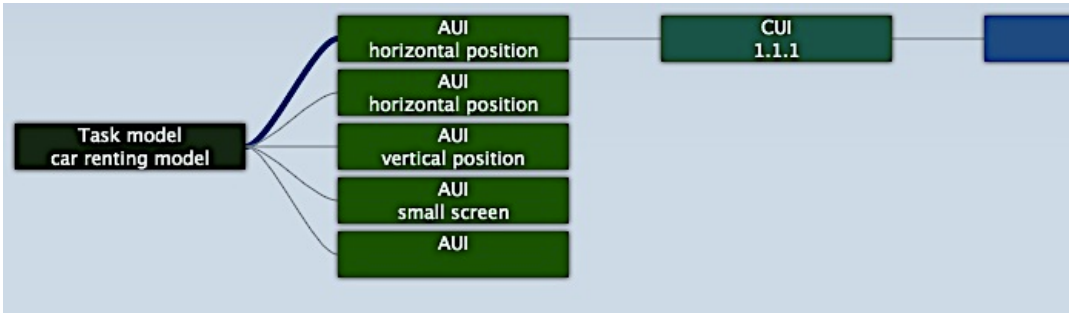


Figure 73. Model Voyager: interactive tree of adapted models for the car rental example.

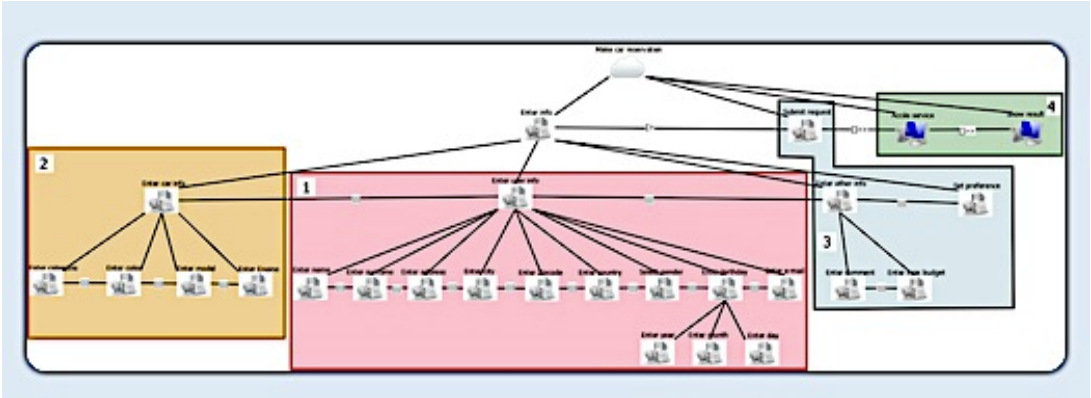


Figure 74. The Abstract User Interface (AUI) for the car rental example.

The Figure 73 and Figure 74 respectively present the task tree for the models and the Abstract User Interface of the car rental example. The Model Voyager can be used to retrieve several examples of Task models, AUI, CUI and FUI that complement the examples presented in this chapter.

Chapter 5 Evaluation

A central research question is *how* to evaluate a framework. One possible approach has been to apply the framework in different situations, i.e. considering different types of projects, industrial and scientific domains, different application domains and different complexity levels [Bar05] by means of case studies, as Chapter 4 reports. A complementary approach to assess a framework's utility consists in fitting several published works by practicing researchers to frameworks in the studies, as propose Scholtz and Consolvo (2004). In our context the literature search, Chapter 2 reports, enabled to identify and to select concepts that are relevant for CAA and as such essential to be considered in the definition of TriPlet components.

TriPlet has also been defined within the context of the Serenoa Project, therefore, all its development steps have been deeply discussed and analyzed by the project partners and reviewers, who are experts in the domain of CAA, with perspectives from the industry, academia and standardization bodies (W3C), and different profiles, as developers, project manager, researchers, UI designers. This iterative evaluation of TriPlet, although empirical, has been relevant for critically analyzing the decisions about the framework and to achieve better results, i.e. synchronizing it with actual users' needs, perspectives and requirements.

To validate TriPlet, four methods have been combined: first a literature search (as described in Chapter 2) identified essential requirements, then two case studies lead to several instantiations of the framework, testing TriPlet's suitability and applicability for different scenarios (as Chapter 4 describes).

Table 11 locates TriPlet according to the concepts that are commonly found in works about adaptation.

Table 11. TriPlet frameworks classified according to the context (user, platform and environment), support provided and system aspects (presentation, navigation and content). From – non-existing, + low, ++ middle, to +++ high.

	Context			Support	Aspect		
	User	*Plat*	*Env*	*Type*	*Pres*	*Nav*	*Con*
TriPlet	+++	+++	+++	Meta Model, Reference Framework, Adaptation Techniques, Design Space	+++	+++	+++

By analyzing Table 11 we notice that due to the extensive literature review, TriPlet successfully covers multidimensional aspects of adaptation, i.e. the 3 main dimensions of context: user, platform, and environment, the 3 main system aspects: presentation, navigation, and content, and also provides 4 different solutions for stakeholders: a meta model, a reference framework, adaptation techniques, and a design space.

To complement the analysis of TriPlet, in the following sections, we present a static analysis that assess the TriPlet components according to a set of criteria; such analysis is also followed by a set of respective lessons learned.

According to Zelkowitz and Wallace (1998), a static analysis can be performed to characterize the project results and derive also lessons learned, i.e. qualitative data from completed projects that can aid to determine trends, successful approaches and improvement points. Aiming to formally define an evaluation process for the static analysis, we firstly selected specific criteria to analyze the framework, then we conducted a static analy-

sis to evaluate relevant aspects of the framework. Finally we discuss the results obtained and the lessons learned.

5.1 Criteria

Several criteria can be applied to evaluate different quality aspects of a framework. For this project, we selected 8 criteria that according to the literature review have been judged as important in the research domain of CAA:

- **Understandability:** concerning the notation employed to present the diagrams of the framework, if they are also comprehensible, and readable in different formats, notations, modalities for constrained scenarios;
- **Extensibility:** the ability to extend it to include novel technologies and contextual concepts that will be continuously launched in the future;
- **Flexibility:** the ability to be flexible, versatile, supporting different applications, complexity levels, technologies, etc.;
- **General-purpose:** how generic the framework is, i.e. it can be applied regardless of the application domain of the system to be, the contextual information in use, the system aspects affected by the adaptation, and the development approaches and design decisions chosen;
- **Modifiability:** support for adding, deleting, modifying or varying concepts;
- **Scalability:** the ability to scale the framework, i.e. to apply it to more complex applications that involve not only several contextual information but also several adaptation techniques;
- **Usability:** the ability to easily, efficiently and effectively apply the framework (e.g. how intuitive, how difficult it is to quickly find resources, information of interest, to benefit from TriPlet solutions);
- **Utility:** the usefulness of such a framework, how it suits to different projects, scenarios and stakeholders.

5.2 Static Analysis

Given the set of 8 criteria defined in Section 5.1, a static analysis was performed concerning each component of TriPlet. The results of such analysis are reported in Table 12.

Table 12. Static analysis of TriPlet components.

	CAMM	CARF	CADS
Understandability	It has been implemented with MOF, it has a graphical and a textual notation (XML-based) which are convertible to other formats	It has a graphical representation but also an XML-based version, convertible too	It has a graphical representation, editable as ppt and also an HTML5 definition

Extensibility	It can be extended (or refined) if necessary	Except for its central core, all the branches can be expanded covering more concepts	It can be extended by adding new dimensions or refining the existing ones
Flexibility	It covers from simple to complex adaptation applications	It enables finer (or coarser) grained analysis	It enables finer (or coarser) grained analysis
General-purpose	It is meant to accommodate many different application domains	Several different application types can be analyzed	Several different application types can be analyzed
Modifiability	It enables its partial application by using only mandatory concepts	It supports changes in its concepts, e.g. by adding new ones or removing them	It enables to vary its concepts, e.g. specializing, refining
Scalability	Once all adaptation steps are covered, highly complex applications can rely on the concepts already defined in the meta-model	It scales as much as necessary once it supports an extensive list of concepts	Supports the analysis of complex applications, but in one diagram a limited set of instances can be jointly compared; various graphics must be used assuring legibility
Usability	It is accessible as a graphic representation, as a source file (or XML-based version)	The mind maps can be exported in a XML-based format, or graphically edited using FreeMind	The standard diagram must be augmented with the instance layer
Utility	It provides an overview of all concepts relevant for CAA, their properties and associations too	It serves a twofold purpose: taking (new) or analyzing (previous) design decisions	It is useful for analyzing, comparing and evaluating adaptation levels

5.2.1 TriPlet Scalability

While we assume that the 3 components of TriPlet could scale well for complex applications that involve both several contextual information and several adaptation techniques, further studies and experiments would be necessary to actually measure the coverage of the CAMM, CARF and the CADS.

In principle the definition of CAMM, by relying in a set of previous works on the domain of interest, already cover all essential concepts for adaptation processes, as well as their relationships, attributes and methods. However, in a complementary approach for evaluation, by external tests in the industry and with stakeholders could help to identify concepts that are missing.

The CARF has also been built on top of works previously published in the domain of interest. Although it covers an extensive list of concepts, e.g. involving more than 150 techniques, it must be subject to continuous updates to follow the novel trends of technology and adaptation. The mind map diagram scales well, supporting new branches and leaves.

Concerning the scalability of the CADS the main trade off found is potential readability issues, i.e. when several applications must be analysed simultaneously, the diagrams must be be co-located, avoiding legibility problems. Actually we assume that the threshold is given by 3 applications at a time. More than this could lead to confusing interpretations of the graphical representation. In spite of CADS covering 8 dimensions within the same graphical view, we believe that finer grained analysis are suitable as well, i.e. by selecting a set of criteria of interest for evaluation. However, if more than 8 dimensions are considered, this could lead to readability issues, specially concerning printed versions of the diagram.

5.2.2 Discussion

So far, to access TriPlet components users rely in a centralized source, i.e. with an online application, users retrieve the source documents online in the webpage (sites.uclouvain.be/mbui/caa/).

All components can be extended, in a scalable approach, however their main principles and purposes must be respected. While for CADS the dimensions must follow a certain order, and be able to be compared in a consistent manner, for CARF the central branches can be instantiated with additional concepts only if the original definitions are respected.

For UI designers, i.e. the target audience of TriPlet, to benefit of the framework components, a tutorial may be necessary, since the stakeholders can be not familiar with terminology employed. In principle, we could assume that it is enough to read the specification of the framework, to understand, to use and to apply it, however it is unknown how much efforts are indeed necessary in this activity, i.e. how is the learnability of the TriPlet framework.

For the complete versions of the CAMM, CARF and the CADS some legibility issues can occur, i.e. a digital (online) version of the diagrams is more accessible, and scale better, however for printed versions, care must be taken to avoid readability issues.

5.3 Lessons Learned

For implementing and using CAMM the main lesson learned is that several different approaches are possible to implement and to model CAA processes, so only by having a set of quality criteria well-defined since the beginning of the project, a trade-off analysis has been possible. It facilitates the decision making process among several potential solutions.

For CAMM's definitions, MOF, XSD Schema, and Visual Paradigm have been adopted. Specific criteria guided the choice of these formats, like: adoption, popularity, standards, interoperability, robustness, expressiveness, flexibility, and tool support. Visual paradigm supported all features needed to implement the model, including the automatic generation of the XSD Schema.

For creating the CARF, it has been a good decision to define the methodological approach since the beginning: a systematic review to structure the Cards template. The fields selected to compose the Cards also proved to be enough comprehensive for its original purpose. However it could have been better to define also since its start a more flexible notation to be adopted to express it, and continuously publishing it online in a public space, since several authors requested it during the project (while it was not yet publicly available). This approach would also enable the outcomes to be continuously subject to evaluation, and the costs of transferring it after the end of the project could have been reduced. Moreover it is not feasible to be exhaustive, so the progress of CARF needs to be continuously ensured in a collaborative approach. For instance, by means of an online tool with authentication methods to provide access to it, and also to enable collaborative validation of the new contents added. An online wiki page open for the public for search and retrieval seems to be the best solution for the CARF, being open for collaboration and subject to peer-to-peer (voluntary) validation.

For CADS, the radar chart has the right features to express the information of interest (coverage levels and consistent dimensions), however an interactive tool would leverage its usage and make it more flexible and easier to use. Additional features as export, print or customize could facilitate its adoption, meet stakeholders' requirements, and provide more interaction. This same CADS diagram can be used in several domains and also for other purposes (e.g. assessing UX quality criteria).

Table 13 summarizes the lessons that have been learned with TriPlet components: CAMM, CARF and CADS, summarizing its main success factors and decisions that could lead to an improved result.

Table 13. Lessons Learned.

	Success Factors	Improvements
CAMM	It covers extensively the adaptation process, all SDLC phases, it accommodates different instantiations by providing conceptual and abstract solutions	Maybe an eCore model could give more flexibility to generate a language and an editor, it should be subject to more validation by the community (target audience)
CARF	It covers broadly several design decisions for CAA, supporting all development lifecycle by means of an extensive catalog of design decisions, it enables continuous progress	An online collaborative wiki, with authentication and validation, defined since the beginning of the project would facilitate its extensibility and continuous growth, enabling also early access and evaluation; a more flexible notation could aid to better manipulate such contents, and to generate varied outcomes and formats
CADS	It is an appealing tool, unified view, provides consistent criteria and levels	A flexible interactive online tool, with export features, would enable more manipulation, interaction, usage and help to spread its adoption

5.4 Project Requirements

The Section 2.5 describes the main shortcomings in the domain of CAA, followed by the respective requirements that could successfully address them. This section discusses the current status of the project concerning each of these requirements.

The terminology adopted must be consistent: the templates specified in the context of CARF defines a consistent terminology, this is also valid for the terms employed in the CADS and the CAMM.

R1. **Multidimensional context of use.** The context information cannot be limited to one, two or three dimensions, it must be not only broadly considered but complete and also enable extensions;

R2. **Multidimensional system aspects.** CAA cannot target to specific sub-properties of applications, the integral application, concerning navigation, presentation and contents, as well as their specific granularity levels must be carefully considered;

R3. **Application domain independency.** Once all application domains can benefit of CAA, theoretical methods that support CAA must be able to accommodate several scenarios;

R4. **Complete SDLC Coverage.** Methods must support adaptation in the entire lifecycle of development (considering, for instance, the feedback from the user to progressively adapt the application);

R5. **High Usability Levels.** End users must have highest priorities, being able to reject, accept, modify and evaluate CAA process. The adaptation engine must be able to evolve by learning with the end users;

R6. **Technology Independency.** The technological spaces cannot be constrained in terms of languages or platforms. The quick evolution of technology must be considered by enabling extensible and flexible approaches;

R7. **Extensible Approach.** The methods must be extensible allowing continuous update of concepts;

R8. **Advanced Logic.** Simple rules can be used as a basis for CAA, however more complex reasoning and inferences must be supported, e.g. with machine learning, ontologies, etc.;

R9. **Unified Vocabulary.** A standard terminology must be defined and largely adopted, resulting in more consistent approaches, enabling and facilitating the re-use and extensions.

Concerning the current status of TriPlet, the requirements mentioned above, have been addressed to some extent as follows:

R1. **Multidimensional context of use.** TriPlet components consider in an abstract level all dimensions of context (user, platform, environments and application domains). However it was not possible to model all these concepts exhaustively, so we try to ensure enough extensible and a general-purpose to cover a broad range of contextual information;

R2. **Multidimensional system aspects.** The same applies for system aspects, i.e. while classifying them in: content, presentation and navigation enables to cover them broadly, it was not possible to be exhaustive due to time constraints. Moreover new UI elements may appear, so in TriPlet development we try to cover a broad range of system aspects, however just by leaving room for extensions, new elements can be progressively included;

R3. **Application domain independency.** In principle, due to the fact that the TriPlet components cover conceptual and abstract concepts of CAA, they are domain-independent. Although we assume that TriPlet has a generic-purpose in this sense, the validation phase has specifically covered a car rental and a touristic application example;

R4. **Complete SDLC Coverage.** TriPlet support stakeholders in all SDLC, however it specially contributes during design and analysis phases, the implementation is not support and neither the evaluation per se;

R5. **High Usability Levels.** Although TriPlet assumes that CAA provides benefits for the end user, no validation has been carried in this sense, so further efforts are needed to state to which extent TriPlet can actually help stakeholders to achieve higher usability levels in their applications;

R6. **Technology Independency.** The TriPlet components are not constrained in terms of languages or platforms. Given its abstract, conceptual extensible and flexible definitions, it can be applied regardless of the technology employed;

R7. **Extensible Approach.** TriPlet components are in principle extensible and enable continuous updates. However it is not known how much training may be needed for stakeholders to understand, to use and to extend it. Moreover currently the web page is under development, and no features are available for registering new admin and user, there is no authentication features, neither the validation of new entries.

R8. **Advanced Logic.** Although CARF provides a set of adaptation techniques that can be combined to support more complex reasoning and inferences for CAA, there is no algorithm or solution proposed in this sense. CAMM structures the definitions of rules, strategies and policies that abstract and prioritize the CAA processing, however no concrete solution is proposed for implementations;

R9. **Unified Vocabulary.** TriPlet components may aid to reach a standard vocabulary for CAA. However for it to be largely adopted, further efforts are needed, one possible initiative could be publishing its definitions as a W3C document. Still, it is quite impossible to ensure consistency in a global perspective of SDLC.

Figure 75 illustrates an empirical analysis about the current coverage of the requirements defined for this thesis and presented in Section 2.5. On one hand, the multidimensionality of context (R1), system aspects (R2), domain-independency (R3), and technology independency (R6) clearly benefits from the following qualities of the TriPlet components: the conceptual approach, the abstract definitions and the general-purpose, flexible and extensible perspectives. On the other hand, it does not cover the complete SDLC of CAA implementation (R4), mainly because the implementation per se is left for the developers, as well as the decisions regarding the reasoning engine for adaptation that could provide advanced logic (R8) and the usability evaluation of the system (R5).

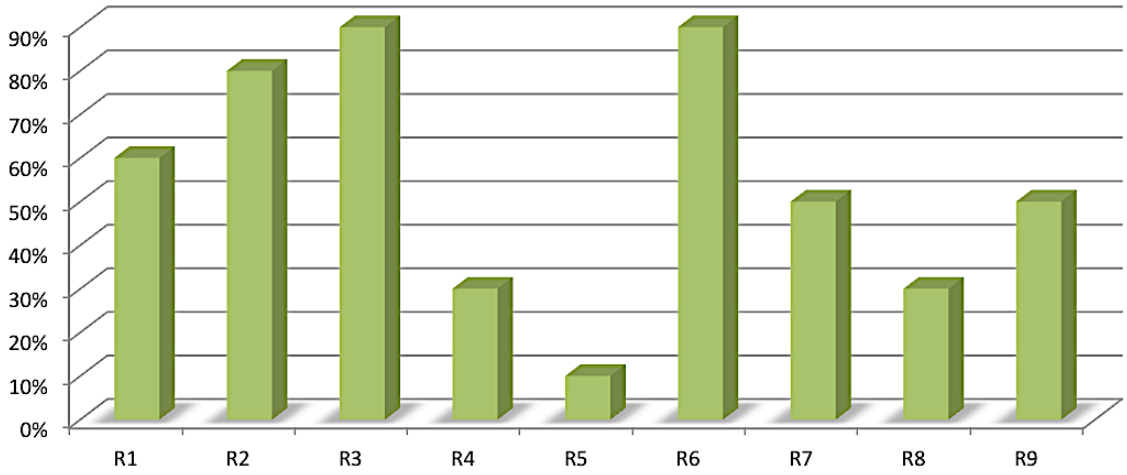


Figure 75. Requirements coverage.

Actually, concerning R4 we can assume that TriPlet aids the initial phases of development (definitions, requirement gathering, design decisions), but the implementation, tests and evaluation are left for the stakeholders to decide. Some initial work[17] has been done to support additional design decisions for the implementation phase. As the Figure 88 in the Appendix K shows, a decision table has been constructed to guide stakeholders in finding appropriate technologies (such as programming, scripting and markup languages) and also methodological approaches that support CAA to implement their applications. The decisions are taken according to the contextual information targeted by the application requirements.

Due to the fact that the TriPlet components are not (yet) popularly employed, it does not reach a broad community that could significantly contribute to advance its definitions (R7), even though the solutions are enough flexible to accommodate extensions, no validation has been done in this sense, to ensure that all concepts are clearly understandable, the users can be registered, provide their own contributions, and validate the existing ones.

[17] Such work has been done thanks to the support of Denis Cariat, in the scope of his master thesis [Car13].

Concerning the unified vocabulary (R9), TriPlet advance in establishing a common terminology for CAA, however only by disseminating it largely, we could expect to have a broad adoption and more consistent results for CAA processes.

5.5 Final Remarks

As mentioned at the beginning of this chapter, currently, frameworks have been evaluated by means of generating several instances of it, for multiple application scenarios. This scenario is useful to illustrate the application of TriPlet. By instantiating its components in varied samples, we could have a clearer idea about its use, mainly concerning some specific criteria, such as its feasibility and usability. So far, the TriPlet components were validated in terms of applicability and evaluated according to a set of selected criteria.

Regarding the requirements initially defined for this thesis, we note that overall most of them have been met. However because of an evident trade-off between the quality of general-purpose and technological support, four requirements (R1, R2, R3 and R6) have been prioritized. So, while TriPlet provides solutions that are multidimensional in terms of: context, system aspects, domain independency and technology independency, the stakeholders must decide regarding the advanced logic (R8) and uncovered phases of the SDLC (R4), as tests and usability evaluation (R5). Besides this, only by reaching a broad, or global community the extensibility (R7) and the consistency (R9) of the concepts could be ensured.

Chapter 6 Conclusion

It is a challenge to identify all context information relevant for adaptation considering multiple dimensions. But it is even more challenging to consider this information and to provide users adaptation with high usability levels and transparency. An excessive use of adaptive techniques can confuse end users, get them lost during the navigation and cause cognitive overload [Pat99].

Given the relevancy of providing CAA nowadays, in a scenario of device fragmentation, heterogeneous user profiles, and an exponentially growing amount of interactive applications, stakeholders can find several benefits in having a unified framework on which they can rely to develop their applications.

In this sense, the main goal of this thesis has been to define the foundations for developing applications that perform CAA, i.e. by understanding CAA and providing a conceptual framework, stakeholders of such applications can find support for the development phases of CAA. TriPlet provides advantages from different perspectives: *conceptually* it ensures a consistent terminology and approach for CAA for different authors; *methodologically* it facilitates the re-use of previous works in the domain of interest, allowing further extensibility of applications, by means of more flexible and compatible approaches; *empirically* it unifies CAA solutions; and *pragmatically* it offers a standard framework that can be universally adopted for implementing CAA.

6.1 Main Contributions

As this thesis statement defines, this PhD project proposes, defines, develops, and instantiates a multidimensional conceptual framework (named TriPlet). TriPlet provides a meta-model (CAMM), a reference framework (CARF), and a design space (CADS), that support stakeholders with structured guidance for addressing context-aware adaptation of user interfaces.

The contributions of this thesis are the result of an extensive and systematic review and analysis of the scientific literature regarding CAA. Such review resulted in the following specific contributions:

- A collection of works have been analyzed covering different application domains, systems aspects and contextual information for CAA, their adaptation rules have been extracted and Cards have been generated based on the adaptation techniques identified:
  - A set of models (14) for adaptation have been selected, retrieved and analyzed, their fundamental concepts have been compared, discussed and identified;
  - A set of frameworks (21) for CAA have been identified, analyzed and discussed;
  - A set of design spaces (12) for CAA have been selected, analyzed and discussed.

The SLR enabled us to extract fundamental concepts for CAA. They provided the ground information to define a conceptual framework for CAA. This conceptual framework, named TriPlet, supports stakeholders during the SDLC of CAA, from design until analysis. TriPlet provides 3 main contributions:

- The CAMM implemented in MOF that formalizes and abstract the main concepts of CAA (adapters, context, adaptation, and models), as well as their main attributes, methods and associations. CAMM provides a unified approach and a consistent terminology for CAA, and it has an associated XML Schema that expresses its definitions enabling validation, documentation and code generation.
- The CARF provides a catalog for stakeholders containing different examples of potential design decisions for implementing CAA. It includes their definitions and main specifications. The *how* branch in the CARF specially includes:
  - 154 descriptive cards composed by 11 fields that precisely defines adaptation techniques and that can be used as design patterns for CAA;
- The CADS supports stakeholders in analyzing, comparing and evaluating 8 coverage levels of adaptation for context-aware applications.

These three main components of TriPlet provide contributions that are: domain-independent, modality-independent and technology-independent. TriPlet is also extensible, flexible and covers a general-purpose.

6.2 Validation of Results

TriPlet suitability has been continuously evaluated by means of 4 validation approaches. First, during its definition phases, a *literature search* has been conducted, to identify main gaps in the domain and progress the current state-of-the-art. Then, TriPlet has been created. Its contributions have been iteratively discussed with experts in the domain (within the scope of the Serenoa project during periodic meetings with reviewers and experts from academia, industry and standardization bodies). Therefore all the intermediary proposals and design decisions have been critically analyzed and largely discussed with the project partners and scientific reviewers. Then, the *case studies* have been defined and several implementations have been built using TriPlet. Such implementations have been decided to explore the general-purpose of the solution, i.e. by considering vastly both the context information (user, platform and environment), and the respective adaptation techniques (impacting content, presentation and navigation). Finally, the results of the case study enabled a *static analysis* (based on a set of quality criteria) and to identify the *lessons learned*.

6.3 Scope

Context-aware adaptation involves several disciplines of computer science, as distributed systems, ubiquitous computing, software engineering and architecture. Aiming to achieve high usability levels and to bridge the gap between users' actual needs and what CAA provides, the main focus of this project is targeted in its Human-Computer Interaction aspects. Although CAA covers a broad range of concepts, the validation cannot be exhaustive, therefore the implementations selected for the case studies cover a variety of aspects of CAA, assuming in principle that alternative implementations could also be supported by TriPlet (by generalization). For the sake of simplification, though, the implementations mainly cover: web applications and graphical user interfaces.

6.4 Limitations

Most of the related works consider a web-based context, although this fact can be considered as a limitation of the project, we assume that the contributions of this thesis are generic enough and flexible to also accommodate applications that are not web-based i.e. still the contextual information is a requirement to define appropriate adaptation and its retrieval is usually facilitated with mobile devices, however given the different granularity levels of the adaptation techniques, they can be extrapolated to be applied in a scenario of native applications too.

Even considering an extensive list of related works and possible concepts, we are aware that not all possible approaches are considered; as such we try to be as extensible to accommodate future developments and also flexible enough to permit alternative solutions to be incorporated.

The conceptual framework aimed in this thesis is not meant to provide an adaptive application for stakeholders or a COTS (commercial-off-the-shelf) solution, but to provide guidance for the implementation of adaptive and adaptable applications in all the different phases of their software development lifecycle.

TriPlet also does not provide a technological environment for implementing applications, as an authoring tool or an IDE, however such environments could be defined based on the contributions of TriPlet since several aspects of CAA have been covered and specified.

6.5 Exploitation

As reported also in [Mag12] and [Mot13b], we do agree that in spite of current work practices are based on a stable environment for interaction (good lighting conditions, silence, static), currently the interaction with mobile devices takes place in varied environments with dynamic conditions of light and noise. In this sense, the industrial practitioners could certainly benefit of tools that facilitate the design and the development of adaptive and adaptable applications, and the adoption of responsive designs.

TriPlet could be incorporated into industrial practice as:

- **context cards:** to illustrate which and how adaptation techniques are suitable for user interfaces, as Figure 76 illustrates; these cards could bring information about actual contexts of use into daily work practices, improving responsiveness and quality in the resulting designs, as also suggested in the work of [Mag12];
- **an authoring tool:** as proposed in [Mot13c], the adaptation concepts, once incorporated in a design assistant environment could guide designers and developers by matching information from the target context of use and the UI components,; this tool by suggesting adaptation guidelines could lead to applications of higher quality, e.g. concerning accessibility and responsiveness. Such a tool could benefit from a mashup approach, by dragging and dropping modules that combine web services providing adaptation functionalities;

♦ **a design pattern language:** the taxonomy as CAMM defines by means of a meta-model for adaptation, could lead to a language definition. In the industrial context, a design pattern language could guide the design and development of applications that consider CAA;

♦ **an analysis tool:** the visual representation of CADS could aid stakeholders to analyse the level of coverage of adaptation in their projects, aiding the location of underexplored regions and suggesting thus future venues for extensions.

Tools as the context cards and an authoring environment could help to achieve applications that properly provide adaptation, matching target contexts with existing techniques. These tools would not only reduce the development time, once the techniques are no longer in scattered sources, but also ensure higher quality levels, by respecting well-known principles and guidelines.

6.6 Final Remarks

It is clear that working with CAA in a broad perspective risks several trade-offs (e.g. performance, privacy, etc.) and also represents a great challenge. According to Koch [Koc01] *"Very formal methods have the advantage to allow correctness proofs, but they add many formalisms that tend to abort creativity. Formless development is chaotic and seldom conducts to a successful project."*. Because a unified view on CAA is the most considerable gap in this domain, we highlight the importance of working in a broad perspective to tackle the main issues in this domain. Besides this, in order to conduct the research in a reasonable manner and aware of the possible issues, the methodology of the project has been defined since its beginning, systematic methods have being adopted (e.g. SLR), in order to organize and document relevant concepts in a more structured approach. The case studies selected focus on heterogeneous application scenarios to effectively validate the outcomes benefits. Although generalization is assumed, further investigation and experiments are needed to effectively prove it.

6.7 Future Works

Given the current contributions of this project, some future venues are possible.

♦ An environment for authoring CAA applications could be built based on the specifications provided by TriPlet components:

- the meta-model defined by CAMM can be translated into other formats as eCore models to generate graphical editors that are compatible with the CAA definitions;
- the schema defined by CAMM can be used for generation of further applications by instantiations, for validation of existing ones, and also for support and document applications that consider CAA;
- the CAMM vocabulary can be used to generate a specific language that enable the implementation of applications that support CAA;

- the techniques for adaptation can be implemented in a web based language, enabling composition of services, generation of mashup applications that access them (such services can be also provided as browser extensions, as plugins or add ons);

The techniques described in the cards (Figure 76) can also lead to the definition of a pattern language, since their goals converge, i.e. both aim at detailing good design practices within a field of expertize. Each technique reported in a card can lead to an 'adaptation pattern' aiding stakeholders to solve design problems according to their target context of use.

- An online repository that extensively covers CAA is envisaged, containing:
  - the CARF as a publicly available, online, and interactive tool;
  - the CARF Cards presenting adaptation techniques to be used as design patterns for CAA;
  - the CARF Cards available for printing, based on different stylesheets, to be used as support material for guiding design sessions (Figure 76);
  - the CADS available in a more flexible version of interactive tool (in which stakeholders set their coverage criteria, dimensions and levels, and the tool automatically generates the diagram, e.g. as an image or a printed file), with additional features (e.g. customization, export);
- Experimental analysis can be planned and conducted to evaluate the qualities of TriPlet, e.g. its acceptance, learnability, and understandability, among its target audience (UI designers).

With a broad dissemination and adoption of the TriPlet components, by means of portal, researchers or practitioners interested in retrieving information in the domain of CAA, can search for, access and retrieve related information, contents, as technologies, tools and current solutions that can be made publicly available online.

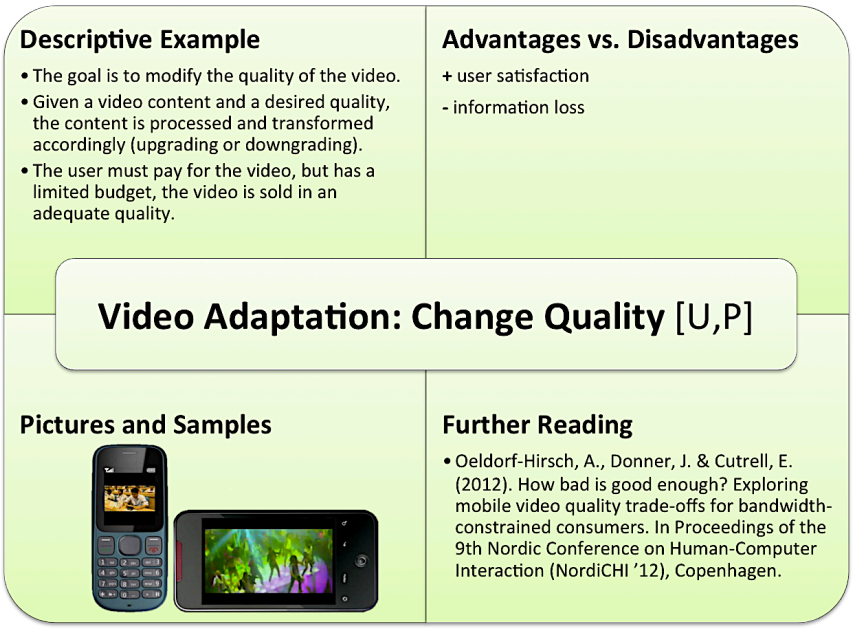


Figure 76. A Card example for video adaptation that can be used to support UI designers during design sessions.

My Publications

Full Papers

Vivian Genaro Motti, Jean Vanderdonckt. 2013. A Unified Model for Context-aware Adaptation of User Interfaces. In Revista Romana de Interactiune Om – Calculator. Volume 6. N. 3. ISSN 1843-4460. 211-248.

Vivian Genaro Motti, Dave Raggett, Sascha Van Cauwelaert, and Jean Vanderdonckt. 2013. Simplifying the development of cross-platform web user interfaces by collaborative model-based design. In Proceedings of the 31st ACM international conference on Design of communication (SIGDOC '13). ACM, New York, NY, USA, 55-64. DOI=10.1145/2507065.2507067 http://doi.acm.org/10.1145/2507065.2507067

Vivian Genaro Motti, Dave Raggett, Jean Vanderdonckt: Current Practices on Model-based Context-aware Adaptation. In Proceedings of CASFE at EICS 2013: 17-23

Vivian Genaro Motti, Jean Vanderdonckt. 2013. A Computational Framework for Context-aware Adaptation of User Interfaces. RCIS'2013. In Proceedings of the 7th IEEE International Conference on Research Challenges of Information Science.

Ugo Braga Sangiorgi, Vivian Genaro Motti, François Beuvens, and Jean Vanderdonckt. 2012. Assessing lag perception in electronic sketching. In Proceedings of the 7th Nordic Conference on Human-Computer Interaction: Making Sense Through Design (NordiCHI '12). ACM, New York, NY, USA, 153-161. DOI=10.1145/2399016.2399040 http://doi.acm.org/10.1145/2399016.2399040

Charles-Eric Dessart, Vivian Genaro Motti, and Jean Vanderdonckt. 2012. Animated transitions between user interface views. In Proceedings of the International Working Conference on Advanced Visual Interfaces (AVI '12), Genny Tortora, Stefano Levialdi, and Maurizio Tucci (Eds.). ACM, New York, NY, USA, 341-348. DOI=10.1145/2254556.2254623 http://doi.acm.org/10.1145/2254556.2254623

Charles-Eric Dessart, Vivian Genaro Motti, and Jean Vanderdonckt. 2011. Showing user interface adaptivity by animated transitions. In Proceedings of the 3rd ACM SIGCHI symposium on Engineering interactive computing systems (EICS '11). ACM, New York, NY, USA, 95-104. DOI=10.1145/1996461.1996501 http://doi.acm.org/10.1145/1996461.1996501

Short Papers

Vivian Genaro Motti, Ugo Sangiorgi, and Jean Vanderdonckt. 2013. Enhancing collaborative sketching activities with context-aware adaptation guidelines. In Proceedings of the 19th Brazilian symposium on Multimedia and the web (WebMedia '13). ACM, New York, NY, USA, 149-152. DOI=10.1145/2526188.2526220 http://doi.acm.org/10.1145/2526188.2526220

Ugo Braga Sangiorgi, Mathieu Zen, Vivian Genaro Motti, Jean Vanderdonckt: Challenges on Distributing a Collaborative Sketching System Across Multiple Devices. In Proceedings of Distributed User Interfaces (DUI) at EICS 2013: 50-53

Vivian Genaro Motti, Ugo Braga Sangiorgi, Jean Vanderdonckt: Enhancing Collaborative Sketching with Adaptation Guidelines. In Proceedings of CASFE at EICS 2013: 1-4

Vivian Genaro Motti and Dave Raggett. 2013. Quill: a collaborative design assistant for cross platform web application user interfaces. In Proceedings of the 22nd international conference on World Wide Web companion (WWW '13 Companion). International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 3-6.

Vivian Genaro Motti, Nesrine Mezhoudi, Jean Vanderdonckt: A Theoretical Framework for Specifying and Analyzing Context-Aware Adaptation. In Proceedings of CASFE at AmI 2012

Vivian Genaro Motti, Nesrine Mezhoudi, Jean Vanderdonckt: Machine Learning in the Support of Context-Aware Adaptation. In Proceedings of CASFE at AmI 2012

Vivian Genaro Motti. 2011. A computational framework for multi-dimensional context-aware adaptation. In Proceedings of the 3rd ACM SIGCHI symposium on Engineering interactive computing systems (EICS '11). ACM, New York, NY, USA, 315-318. DOI=10.1145/1996461.1996545 http://doi.acm.org/10.1145/1996461.1996545

Tutorials

Vivian Genaro Motti and Jean Vanderdonckt. 2011. Multi-dimensional context-aware adaptation for web applications. In Proceedings of the 11th international conference on Current Trends in Web Engineering (ICWE'11), Andreas Harth and Nora Koch (Eds.). Springer-Verlag, Berlin, Heidelberg, 352-354. DOI=10.1007/978-3-642-27997-3_39 http://dx.doi.org/10.1007/978-3-642-27997-3_39

Vivian Genaro Motti and Jean Vanderdonckt. 2011. Context-aware adaptation of user interfaces. In Proceedings of the 13th IFIP TC 13 international conference on Human-computer interaction - Volume Part IV (INTERACT'11), Pedro Campos, Nuno Nunes, Nicholas Graham, Joaquim Jorge, and Philippe Palanque (Eds.), Vol. Part IV. Springer-Verlag, Berlin, Heidelberg, 700-701.

Workshop

Francisco Javier Caminero Gil, Fabio Paternò, and Vivian Genaro Motti. 2013. Context-aware service front-ends. In Proceedings of the 5th ACM SIGCHI symposium on Engineering interactive computing systems (EICS'13). ACM, New York, NY, USA, 339-340. DOI=10.1145/2480296.2483224 http://doi.acm.org/10.1145/2480296.2483224

Proceedings

Francisco Javier Caminero Gil, Fabio Paternò, Vivian Genaro Motti (Eds.): Proceedings of the Workshop on Context-Aware Adaptation of Service Front-Ends, London, U.K., June 24, 2013. CEUR Workshop Proceedings 1013, CEUR-WS.org 2013

Working Papers

Vivian Genaro Motti, Dave Raggett, Jean Vanderdonckt: Current Practices on Model-based Context-aware Adaptation. LSM Working Papers 2013/15 (April 2013)

Vivian Genaro Motti and Dave Raggett. 2013. Quill: a collaborative design assistant for cross platform web application user interfaces. LSM Working Papers 2013/14 (April 2013)

Vivian Genaro Motti, Ugo Braga Sangiorgi, Jean Vanderdonckt: Enhancing Collaborative Sketching with Adaptation Guidelines. LSM Working Papers 2013/08 (March 2013)

Ugo Braga Sangiorgi, Vivian Genaro Motti, François Beuvens, and Jean Vanderdonckt. Assessing lag perception in electronic sketching. LSM Working Papers 2013/03 (February 2013)

Vivian Genaro Motti, Jean Vanderdonckt. 2013. A Computational Framework for Context-aware Adaptation of User Interfaces. LSM Working Papers 2013/02 (January 2013)

Charles-Eric Dessart, Vivian Genaro Motti, and Jean Vanderdonckt. 2012. Animated transitions between user interface views. LSM Working Papers 2012/29 (December 2012)

Deliverables

Vivian Genaro Motti, Jean Vanderdonckt, Deliverable 2.1.1 Context Aware Design Space and Context Aware Reference Framework (R1) 25/02/2011 FP7 – ICT – 258030 Serenoa. At: http://www.serenoa-fp7.eu/wp-content/uploads/2012/10/SERENOA_D2.1.1.pdf

Vivian Genaro Motti, Jean Vanderdonckt, Deliverable 3.1.1 Reference Models Specification (R1) 30/08/2011 FP7 – ICT – 258030 Serenoa. At: http://www.serenoa-fp7.eu/wp-content/uploads/2012/07/SERENOA_D3.1.1.pdf

Vivian Genaro Motti, Jean Vanderdonckt, Deliverable 4.2.1 Algorithm Library for AAL (R1) 30/09/2011 FP7 – ICT – 258030 Serenoa. At: http://www.serenoa-fp7.eu/wp-content/uploads/2012/10/SERENOA_D4.2.1.pdf

Vivian Genaro Motti, Jean Vanderdonckt, Deliverable 2.1.2 Context Aware Design Space and Context Aware Reference Framework (R2) 29/02/2012 FP7 – ICT – 258030 Serenoa. At: http://www.serenoa-fp7.eu/wp-content/uploads/2012/07/SERENOA_D2.1.2.pdf

Vivian Genaro Motti, Jean Vanderdonckt, Deliverable 3.1.2 Reference Models Specification (R2) 30/08/2012 FP7 – ICT – 258030 Serenoa. At: http://www.serenoa-fp7.eu/wp-content/uploads/2012/08/SERENOA_D3.1.2.pdf

Vivian Genaro Motti, Nesrine Mezhoudi, Jean Vanderdonckt. Deliverable 4.2.2 Algorithm Library for AAL (R2) 31/08/2012 FP7 – ICT – 258030 Serenoa. At: http://www.serenoa-fp7.eu/wp-content/uploads/2012/09/SERENOA_D4.2.2.pdf

Videos

Adaptation for Large Screens – Car Rental Example.
At: http://www.youtube.com/watch?v=YGjEQPRkjv8

Context-aware Adaptation: context of use A
http://www.youtube.com/watch?v=M_7gu5FKFR0

Context-aware Adaptation: context of use B
http://www.youtube.com/watch?v=5hEEmmhpDM8

References

A

[Abo99]

Abowd, G. D. Software engineering issues for ubiquitous computing, Proceedings of the 21st international conference on Software engineering, p.75-84, May 16-22, 1999, Los Angeles, California, United States [doi>10.1145/302405.302454]

[Aca04]

Acay, L. D. (2004) Adaptive User Interfaces in Complex Supervisory Tasks. Master Thesis.

[Alm12]

Almeida, R. A., Pillias, C., Pietriga, E., Cubaud, P. (2012) Looking behind Bezels: French Windows for Wall Displays. In Proceedings of the International Working Conference on Advanced Visual Interfaces (AVI '12), Genny Tortora, Stefano Levialdi, and Maurizio Tucci (Eds.). ACM, New York, NY, USA, 124-131. DOI=10.1145/2254556.2254581 http://doi.acm.org/10.1145/2254556.2254581

[Ard07]

L. Ardissono, R. Furnari, A. Goy, G. Petrone, and M. Segnan, "A framework for the management of context-aware workflow systems," in Proc. of WEBIST 2007 - Third International Conference on Web Information Systems and Technologies, Barcelona, Spain, 2007, pp. 80–87.

[Ard08]

Ardissono, L., Goy, A. and Petrone, G. "A framework for the development of distributed, context-aware adaptive hypermedia applications", in AH'08 Proceedings of the Fifth International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems, W. Nejdl, J. Kay, P. Pu and E. Herder (Eds), Berlin/ Heidelberg: Springer, pp. 259-262, 2008.

[Arh09]

Arhippainen, Leena, Studying user experience: issues and problems of mobile services – Case ADAMOS: User experience (im)possible to catch? Faculty of Science, Department of Information Processing Science, University of Oulu, P.O.Box 3000, FI-90014 University of Oulu, Finland Acta Univ. Oul. A 528, 2009 Oulu, Finland

[Ass07]

Assad, M., Carmichael, D. J., Kay, J. and Kummerfeld, B.. 2007. PersonisAD: distributed, active, scrutable model framework for context-aware services. In Proc. of the 5th international conference on Pervasive computing (PERVASIVE'07), Anthony LaMarca, Marc Langheinrich, and Khai N. Truong (Eds.). Springer-Verlag, Berlin, Heidelberg, 55-72.

B

[Bac10]

Bacha F., Oliveira K., Abed M. (2011). Using Context Modeling and Domain Ontology in the Design of Personalized User Interface. International Journal on Computer Science and Information Systems (IJCSIS), 6, pp. 69-94, ISSN 1646-369.

[Bar03]

Bardram, J.E.: Hospitals of the Future – Ubiquitous Computing support for Medical Work in Hospitals. In: Bardram, J.E., Korhonen, I., Mihailidis, A., Wan, D. (eds.) UbiHealth 2003: The 2nd International Workshop on Ubiquitous Computing for Pervasive Healthcare Applications, Seattle, WA, USA (October 2003), http://www.pervasivehealthcare.dk/ubicomp2003

[Bar05]

Bardram, J. E. (2005) The java context awareness framework (JCAF) – a service infrastructure and programming framework for context-aware applications. In Proceedings of the Third international conference on Pervasive Computing (PERVASIVE'05), Hans-W. Gellersen, Roy Want, and Albrecht Schmidt (Eds.). Springer-Verlag, Berlin, Heidelberg, 98-115. DOI=10.1007/11428572_7 http://dx.doi.org/10.1007/11428572_7

[Bau11]

Bautista, S., Hervás, R., Gervás, P., Power, R., and Williams, S.. 2011. How to make numerical information accessible: experimental identification of simplification strategies. In Proceedings of the 13th IFIP TC 13 international conference on Human-computer interaction - Volume Part I (INTERACT'11), Pedro Campos, Nuno Nunes, Nicholas Graham, Joaquim Jorge, and Philippe Palanque (Eds.), Vol. Part I. Springer-Verlag, Berlin, Heidelberg, 57-64.

[Bic99]

Bickmore, T., Web Page Filtering and Re-Authoring for Mobile Users, The Computer Journal, vol. 42, no. 6, pp. 534-546, Jun. 1999.

[Böh10]

Böhmer, Matthias, Gernot Bauer, and Antonio Krüger. "Exploring the design space of context-aware recommender systems that suggest mobile applications." 2nd Workshop on Context-Aware Recommender Systems. 2010.

[Boh11]

Bohoj, M., Olof Bouvin, N., Gammelmark, H. AdapForms: A Framework for Creating and Validating Adaptative Forms. In Proceedings of International Conference on Web Engineering (ICWE 2011), Paphos, Cyprus. Springer, pages 105-120

[Bre07]

Brereton, P., Kitchenham, B. A., Budgen, D., Turner, M. and Khalil, M.. 2007. Lessons from applying the systematic literature review process within the software engineering domain. J. Syst. Softw. 80, 4 (April 2007), 571-583. DOI=10.1016/j.jss.2006.07.009 http://dx.doi.org/10.1016/j.jss.2006.07.009

[Bre09]

Breiner, K., Gauckler, V., Seissler, M., and Meixner, G.. Evaluation of user interface adaptation strategies in the process of model-driven user interface development Interfaces, 2009.

[Bro86]

Browne, P. D., Sharratt, B. D. and Norman, M. The formal specification of adaptive user interfaces using command language grammar," In CHI'86 Proceedings. April, pp. 256–260, 1986.

[Bru94]

Brusilovsky, P. 1994. Adaptive Hypermedia: An Attempt to Analyze and Generalize. In Selected papers from the First International Conference on Hypermedia, Multimedia, and Virtual Reality: Models, Systems, and Applications (MHVR '94), Peter Brutsilosky, Piet Kommers, and Norbert A. Streitz (Eds.). Springer-Verlag, London, UK, 288-304.

[Bru96]

Brusilovsky P. (1996). Methods and techniques of adaptive hypermedia. User modeling and User- Adapted Interaction. Special issue on: Adaptive Hypertext and Hypermedia, Vol. 6, No. 2-3, July 1996.

[Bru99]

Brusilovsky, P. Adaptive and Intelligent Technologies for Web-based Education. *Künstliche Intelligenz 4* (1999), 19-25.

[Bru01]

Brusilovsky, P.: 2001, Adaptive hypermedia. User Modeling and User-Adapted Interaction 11(1-2), 87-110

[Bru02]

Brusilovsky, P. and Cooper, D. W.. 2002. Domain, task, and user models for an adaptive hypermedia performance support system. In Proceedings of the 7th international conference on Intelligent user interfaces (IUI '02). ACM, New York, NY, USA, 23-30. DOI=10.1145/502716.502724 http://doi.acm.org/10.1145/502716.502724

[Bru07]

Brusilovsky, P., Kobsa, A., and Nejdl, W. The Adaptive Web, Methods and Strategies of Web Personalization Springer 2007

[But07]

Butter, T. Aleksy, M. Bostan, P. and Schader, M. "Context-aware User Interface Framework for Mobile Applications," 27th International Conference on Distributed Computing Systems Workshops (ICDCSW'07), pp. 39–39, 2007.

[Buy00]

Buyukkokten, O., Garcia-Molina, H., Paepcke, A., and Winograd, T. (2000). Power browser: efficient web browsing for pdas. In CHI '00: Proceedings of the SIGCHI conference on Human factors in computing systems, pages 430–437, New York, NY, USA. ACM

C

[Cal02]

Calvary, G., Coutaz, J., Bouillon, L., Florins, M., Limbourg, Q., Marucci, L., Paternò, F., Santoro, C., Souchon, N., Thevenin, D., Vanderdonckt, J., 2002 The CAMELEON Reference Framework, Deliverable 1.1, CAMELEON Project

[Cal03]

Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J. A Unifying Reference Framework for Multi-Target User Interfaces. Interacting with Computers 15, 3 (June 2003), pp. 289-308.

[Cal05]

Calvary, G., Daassi, O., Coutaz, J., & Demeure, A. (2005). Des widgets aux comets pour la Plasticité des Systèmes Interactifs. Revue d'Interaction Homme-Machine, 6(1), 33-53

[Cal07]

Calvary, G. Plasticité des Interfaces Homme-Machine [Report]: Habilitation à Diriger des Recherches / Laboratoire d'Informatique ; Université Joseph Fourier. - Grenoble, 2007

[Cal11]

Calvary, Gaëlle, et al. "Envisioning Advanced User Interfaces for E-Government Applications: A Case Study." Practical Studies in E-Government. Springer New York, 2011. 205-228.

[Car09]

Cardoso, J. C., & José, R. (2009, January). A framework for context-aware adaptation in public displays. In On the Move to Meaningful Internet Systems: OTM 2009 Workshops (pp. 118-127). Springer Berlin Heidelberg.

[Car13]

Cariat, D. "The Rise of Digital Ecosystems: Multidimensional and Context-aware Adaptation of User Interfaces" (2013) Master Thesis – Université catholique de Louvain

[Chu04]

Chu, H., Song, H., Wong, C., Kurakake, S, and Katagiri, M. 2004. Roam, a seamless application framework. J. Syst. Softw. 69, 3 (January 2004), 209-226. DOI=10.1016/S0164-1212(03)00052-9 http://dx.doi.org/10.1016/S0164-1212(03)00052-9

[Com13]

ComScore 'An average Monday in the U.K.' - February 2013 – by Andrew Lipsman and Carmela Aquino. Publicly available at: http://www.comscoredatamine.com/2013/02/an-average-monday-in-the-uk-pcs-for-lunch-tablets-for-dinner/

[Cou02]

Coutaz, J., Rey, G., 2002. Foundations for a Theory of Contextors. In: Kolski, Ch., Vanderdonckt, J. (Eds.), Computer-Aided Design of User Interfaces III, Proceedings of 4th International Conference of Computer-Aided Design of User Interfaces CADUI'2002 (Valenciennes, 15-17 May 2002). Kluwer Academics, Dordrecht, pp. 13–33.

[Cou06]

Coutaz, J. (2006) Meta-User Interfaces for Ambient Spaces. In: Proc. of 4th Int. Workshop on Task Models and Diagrams for User Interface Design Tamo-

dia'2006 (Hasselt, October 23-24, 2006). Lecture Notes in Computer Science, Vol. 4385, Springer, Heidelberg, pp. 1–15.

D

[Deb11]

Debande, F. and Poncelet, Q. "Adaptation du « front-end » d'un web service en fonction du contexte d'utilisation - Concept et Prototype" Master Thesis (2011) – Université catholique de Louvain

[Des10]

Desruelle, H. Blomme, D. and Gielen, F. "Adaptive Mobile Web Applications: A Quantitative Evaluation Approach," In Proc. of: ICWE 2011, LNCS 6757, pp. 375–378, 2011.

[Des11]

Dessart, Ch.-E., Motti, V. G., and Vanderdonckt, J. 2011. Showing User Interface Adaptivity by Animated Transitions. In Proc. of 3rd ACM Symposium on Engineering Interactive Computing Systems. EICS'2011. ACM Press, NY, 95-104.

[Dey00]

Dey, A. K. and Abowd, G. D. Towards a better understanding of Context and Context-Awareness. CHI 2000. Workshop on What, who, Where, When, and How of Context-Awareness (2000).

[Dey01]

Dey, A. K., Abowd, G. D., and Salber, D. 2001. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. Hum.-Comput. Interact. 16, 2 (Dec. 2001), 97-166.

[Die94]

Dieterich, H., Malinowski, U., Kühme, T., and Schneider- Hufschmidt, M. State of the Art in Adaptive User Interfaces, in Adaptive User Interfaces: Principles and Practice, Schneider-Hufschmidt & al. (eds.), 1994, pp. 13-48

[Dir09]

Diraä, N.;, Engelen, J. Ghesquire, P. and Neyens. K. 2009. The Use of ICT to Support Students with Dyslexia. In Proceedings of the 5th Symposium of the Workgroup Human-Computer Interaction and Usability Engineering of the Austrian Computer Society on HCI and Usability for e-Inclusion (USAB '09), Andreas Holzinger and Klaus Miesenberger (Eds.). Springer-Verlag, Berlin, Heidelberg, 457-462. DOI=10.1007/978-3-642-10308-7_33 http://dx.doi.org/10.1007/978-3-642-10308-7_33

[Dit95]

Dittrich, K.R., Gatziu, S., Geppert, A. The Active Database Management System Manifesto: A Rule-base of ADBMS Features. In Proceedings of the 2nd International Workshop on Rules in Database Systems, Vol. 985, Springer-Verlag, 1995, pp. 3-20.

[Doo12]

Doodles. At: http://www.google.com/doodles/about

[Dra11]

Dragicevic, P., Bezerianos, A., Javed, W., Elmqvist, N., and Fekete, J. D. 2011. Temporal distortion for animated transitions. In Proceedings of the 2011 annual conference on Human factors in computing systems (CHI '11). ACM, New York, NY, USA, 2009-2018. DOI=10.1145/1978942.1979233 http://doi.acm.org/10.1145/1978942.1979233

[Dua06]

Duarte, C. and Carriço, L. "A conceptual framework for developing adaptive multimodal applications," Proceedings of the 11th international conference on Intelligent user interfaces - IUI'06, p. 132, 2006.

E

[Eis00]

Eisenstein, J. and Puerta, A. 2000. Adaptation in automated user-interface design. In Proceedings of the 5th international conference on Intelligent user interfaces (IUI '00). ACM, New York, NY, USA, 74-81. DOI=10.1145/325737.325787 http://doi.acm.org/10.1145/325737.325787

[Equ12]

Equation Research on Behalf of Compuware, 2012 'What the Users want from Mobile'. Publicly available at: http://webperformanceguru.files.wordpress.com/2011/07/19986_whatmobileuserswant_wp.pdf

F

[Fah05]

Fahrmair, M., Wassiou S., and Bernd S. "An engineering approach to adaptation and calibration." In Modeling and Retrieval of Context, pp. 134-147. Springer Berlin Heidelberg, 2006.

[Far07]

de Farias, C. R. G., Leite, M. M., Calvi, C. Z., Pessoa, R. M. and Pereira Filho, J. G. 2007. A MOF metamodel for the development of context-aware mobile applications. In Proceedings of the 2007 ACM symposium on Applied computing (SAC '07). ACM, New York, NY, USA, 947-952. DOI=10.1145/1244002.1244209 http://doi.acm.org/10.1145/1244002.1244209

[Fei05]

B. Feiten, I. Wolf, E. Oh, J. Seo, and H.-K. Kim, "Audio adaptation according to usage environment and perceptual quality metrics", IEEE Trans Multimedia, vol. 7, no. 3, pp.446 - 453 , 2005.

[Fis12]

Fischer, G.. (2012). Context-aware systems: the 'right' information, at the 'right' time, in the 'right' place, in the 'right' way, to the 'right' person. In Proceedings of the International Working Conference on Advanced Visual Interfaces (AVI '12), Genny Tortora, Stefano Levialdi, and Maurizio Tucci (Eds.). ACM, New York, NY, USA, 287-294. DOI=10.1145/2254556.2254611 http://doi.acm.org/10.1145/2254556.2254611

[Fos10]

Foss, J. G. K., & Cristea, A. I. (2010). Transforming a Linear Module into an Adaptive One: Tackling the Challenge. (V. Aleven, J. Kay, & J. Mostow, Eds.)Evaluation, 2, 82-91. SPRINGER-VERLAG BERLIN. Retrieved from http://wrap.warwick.ac.uk/5469/

[Fra98]

Frank, M. R. and Szekely, P.. Adaptive Forms: An interaction paradigm for entering structured data. In Proceedings of the ACM International Conference on Intelligent User Interfaces, pages 153-160, (San Francisco, California, January 6-9) 1998.

[Fri09]

Frias-Martinez, E., Chen, S. Y. and Liu, X. Evaluation of a personalized digital library based on cognitive styles: Adaptivity vs. adaptability, International Journal of Information Management, vol. 29, pp. 48-56, 2009.

[Fuc05]

Fuchs, F., Hochstatter, I., Krause, M., and Berger, M. 2005. A Metamodel Approach to Context Information. In Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOMW '05). IEEE Computer Society, Washington, DC, USA, 8-14. DOI=10.1109/PERCOMW.2005.9
http://dx.doi.org/10.1109/PERCOMW.2005.9

G

[Gaj04]

Gajós, K. and Weld, D. S.. SUPPLE: Automatically Generating User Interfaces. In Proceedings of IUI'04. Funchal, Portugal, 2004

[Gaj06]

Gajós, K. Z., Czerwinski, M., Tan, D. S., and Weld, D. S.. 2006. Exploring the design space for adaptive graphical user interfaces. In Proceedings of the working conference on Advanced visual interfaces (AVI '06). ACM, New York, NY, USA, 201-208. DOI=10.1145/1133265.1133306 http://doi.acm.org/10.1145/1133265.1133306]

[Gaj08]

Gajós, K. Z., Everitt, K., Czerwinski, M., Tan, D. S. and Weld, D. S. Predictability and accuracy in adaptive user interfaces. In Proceedings of CHI'08. Florence, Italy, 2008. CHI Note.

[Gan07]

Ganneau, V., Calvary, G., and Demumieux, R.. 2007. Métamodèle de règles d'adaptation pour la plasticité des interfaces homme-machine. In Proceedings of the 19th International Conference of the Association Francophone d'Interaction Homme-Machine (IHM '07). ACM, New York, NY, USA, 91-98. DOI=10.1145/1541436.1541454 http://doi.acm.org/10.1145/1541436.1541454

[Gin12]

Ginige, A., Romano, M., Sebillo, M., Vitiello, G. and Di Giovanni, P. 2012. Spatial data and mobile applications: general solutions for interface design. In Proceedings of the International Working Conference on Advanced Visual Interfaces (AVI '12), Genny Tortora, Stefano Levialdi, and Maurizio Tucci (Eds.). ACM, New York, NY, USA, 189-196. DOI=10.1145/2254556.2254591 http://doi.acm.org/10.1145/2254556.2254591

[Góm09]

Gómez, J. M. and Tran, T.: 2009, A Survey on Approaches to Adaptation on the Web. In: Lytras, M.D., Ordóñez de Pablos, P. (Eds.), Emerging Topics and Technologies in Information Systems. pp. 136-152, IGI Global Publishing, Hershey. In: Miltiadis D. Lytras (Ed.) (The American College of Greece, Greece).

[Gon11]

Goncu, C. and Marriott, K.. 2011. GraVVITAS: generic multi-touch presentation of accessible graphics. In Proceedings of the 13th IFIP TC 13 international conference on Human-computer interaction - Volume Part I (INTERACT'11), Pedro Campos, Nuno Nunes, Nicholas Graham, Joaquim Jorge, and Philippe Palanque (Eds.), Vol. Part I. Springer-Verlag, Berlin, Heidelberg, 30-48.

[Gro05]

Grolaux, D., Vanderdonckt, J., Van Roy, P., Attach me, Detach me, Assemble me like You Work, Proc. of INTERACT'2005 (Rome, 12-16 September 2005), M.-F. Costabile, F. Paternò (eds.), Lecture Notes in Computer Science, Vol. 3585, Springer- Verlag, Berlin, 2005, pp. 198-212.

[Gue11]

Guerreiro, T., Oliveira, J., Benedito, J., Nicolau, H., Jorge, J. and Gonçalves, D. 2011. Blind people and mobile keypads: accounting for individual differences. In Proceedings of the 13th IFIP TC 13 international conference on Human-computer interaction - Volume Part I (INTERACT'11), Pedro Campos, Nuno Nunes, Nicholas Graham, Joaquim Jorge, and Philippe Palanque (Eds.), Vol. Part I. Springer-Verlag, Berlin, Heidelberg, 65-82.

H

[Han04]

Hanumansetty, G. Reena. (2004) Model Based Approach for Context Aware and Adaptive User Interface Generation. Master's Thesis, Virginia Polytechnic Institute and State University, Falls Church, Virginia, USA.

[Hea12]

Heath, T. B. 2012, Advertising based on environmental conditions, US Patent 8138930.

[Hoo97]

Hook K. (1997). Evaluating the Utility and Usability of an Adaptive Hypermedia System. Proceedings IUI'97, pp 179-186, Orlando, Florida USA.

[Hor99]

Horvitz, E.: Principles of Mixed-Initiative User Interfaces. Proc. of ACM Conf. on Human Aspects in Computing Systems CHI 1999, ACM Press, New York, 1999, pp. 159-166.

[How05]

Howse, J., Stapleton, G. and Taylor, J. (2005) Spider diagrams LMS Journal of Computation and Mathematics, 8 . pp. 145-194. ISSN 1461-1570

I

[ISO9126]

ISO/IEC 9126-1:2001 Software engineering — Product quality — Part 1: Quality model. At: http://en.wikipedia.org/wiki/ISO/IEC_9126

[ISO9241]

ISO 9241-1:1992. International Organization for Standardization. At: http://en.wikipedia.org/wiki/ISO_9241

J

[Jam03]

Jameson, A. (2003) "Adaptive Interfaces and Agents" In J. A. Jacko & A. Sears (Eds.), Human-computer interaction handbook: Fundamentals, evolving technologies and emerging applications (pp. 305-330). Mahwah, NJ: Erlbaum.

[Jan07]

Jankowska, B. Architectural Frameworks for Automated Content Adaptation to Mobile Devices Based on Open-Source Technologies, PhD Thesis. Europa-Universität Viadrina Frankfurt, Germany, 2007.

K

[Kak10]

Kaklanis, N., Moschonas, P. , Moustakas K. and Tzovaras, D. "Enforcing accessible design of products and services through simulated accessibility evaluation", International Conference on ICT for ageing and eInclusion, CONFIDENCE 2010, Jyväskylä, Finland, December 2010.

[Kan89]

Kantorowitz, E. and Sudarsky, O. The adaptable user interface. CACM, 32(11):1352–1358, 1989. DOI: 10.1145/68814.68820

[Kap03]

Kappel, G., Proll, B., Retschitzegger, W. and Schwinger, W. 2003. Customisation for ubiquitous web applications: a comparison of approaches. Int. J. Web Eng. Technol. 1, 1 (August 2003), 79-111. DOI=10.1504/IJWET.2003.003322 http://dx.doi.org/10.1504/IJWET.2003.003322.

[Kar96]

Karagiannidis, C., Koumpis, A., & Stephanidis, C. (1996, August). Deciding "What","When","Why", and "How" to Adapt in Intelligent Multimedia Presentation Systems. In 12th European Conference on Artificial Intelligence. Budapest, Hungary.

[Kir92]

Kirwan, B. and Ainsworth, L. (Eds.) (1992). A guide to task analysis. Taylor and Francis.

[Kit04]

Kitchenham, B. Procedures for performing systematic reviews. Technical Report, Keele University and NICTA, 2004

[Kob07]

Kobsa, A. "Generic user modeling systems." In The adaptive web, pp. 136-154. Springer Berlin Heidelberg, 2007.

[Kob11]

Kobayashi, M., Hiyama, A., Miura, T., Asakawa, C., Hirose, M., and Ifukube, T.. 2011. Elderly user evaluation of mobile touchscreen interactions. In Proceedings of the 13th IFIP TC 13 international conference on Human-computer interaction - Volume Part I (INTERACT'11), Pedro Campos, Nuno Nunes, Nicholas Graham, Joaquim Jorge, and Philippe Palanque (Eds.), Vol. Part I. Springer-Verlag, Berlin, Heidelberg, 83-99.

[Koc01]

Koch, Nora Parcus de. Software engineering for adaptive hypermedia systems. Diss. PhD Thesis, Verlag Uni-Druck, Munich, 2001.

[Kor12]

Korpi, Joni. Less Framework. Available at: http://lessframework.com/ (2012)

[Kur02]

Kurtev, I., Bézivin, J., and Aksit, M. (2002) Technological Spaces: an Initial Appraisal. CoopIS, DOA'2002 Federated Conferences, Industrial track, Irvine, 2002, http://www.sciences.univ-nantes.fr/lina/atl/www/papers/Position PaperKurtev.pdf

[Kur04]

Kurz, B. Popescu, I. and Gallacher, S. "FACADE - a framework for context-aware content adaptation and delivery," Proceedings. Second Annual Conference on Communication Networks and Services Research, 2004., pp. 46–55, 2004.

L

[Laf00]

Beaudouin-Lafon, M. Instrumental Interaction: An Interaction Model for Designing PostWIMP User Interfaces. In Proceedings of the SIGCHI conference on Human factors in computing systems (CHI 2000), ACM Press, New York, NY, 2000, 446-453

[Lav10]

Lavie, T. and Meyer, J. Benefits and costs of adaptive user interfaces. International Journal of Human Computer Studies, 68 (2010), pp. 508–524.

[Lib11]

Libouton, D. "Un Catalogue des Techniques d'Adaptation d'Interfaces Homme-Machine" (2011) Master Thesis – Université catholique de Louvain

[Lim04]

Limbourg, Q., and Vanderdonckt, J. "Multimodality and context-aware adaptation." In Building the Information Society, pp. 427-432. Springer US, 2004.

[Llo97]

Lloyd R., van Ossenbruggen, J., Hardman, L. and Bulterman, D. C. A.. 1997. A framework for generating adaptable hypermedia documents. In Proceedings of the fifth ACM international conference on Multimedia (MULTIMEDIA '97). ACM, New York, NY, USA, 121-130. DOI=10.1145/266180.266348 http://doi.acm.org/10.1145/266180.266348

[Lóp08]

López-Jaquero, V., Vanderdonckt, J., Montero, F., González, P. Towards an Extended Model of User Interface Adaptation: the ISATINE framework. In: Proc. of EIS'2007. LNCS, Vol. 4940, Springer-Verlag, Berlin, 2008, pp. 374-392.

[Lóp09]

López-Jaquero, V., Montero, F., and Real, F. 2009. Designing user interface adaptation rules with T: XML. In Proceedings of the 14th international conference on Intelligent user interfaces (IUI '09). ACM, New York, NY, USA, 383-388. DOI=10.1145/1502650.1502705 http://doi.acm.org/10.1145/1502650.1502705

[Lóp10]

López Jaquero, V., Montero, F., Sendín, M., González, P. Un Metamodelo para la Especificación de Reglas de Adaptación para Entornos Colaborativos, Proc. of XI Congreso Internacional de Interacción Persona-Ordenador Interacción'2010 (Valencia, 7-9 September), 2010.

[Lor00]

Lorenz, A. Oppermann, R. and Zimmermann, A. "Adaptive and Context-Aware Systems: A Survey." p. 22, 2000.

[Lum02]

Lum, W. Y. and Lau, F. C. M. 2002. A Context-Aware Decision Engine for Content Adaptation. IEEE Pervasive Computing 1, 3 (July 2002), 41-49. DOI=10.1109/MPRV.2002.1037721 http://dx.doi.org/10.1109/MPRV.2002.1037721

[Luy10]

Luyten, K., Haesen, M., Ostrowski, D., Coninx, K., Degrandsart, S., and Demeyer, S.. "On stories, models and notations: Storyboard creation as an entry point for model-based interface development with UsiXML." (2010).

M

[Mag12]

Magnusson, C. Larsson, A. Warell, A. Eftring, H. and Hedvall, P.-O. "Bringing the mobile context into industrial design and development," Proceedings of the 7th Nordic Conference on Human-Computer Interaction Making Sense Through Design - NordiCHI '12, p. 149, 2012.

[Mal10]

Malandrino, D. Mazzoni, F. Riboni, D. Bettini, C. Colajanni, M. and Scarano, V. "MIMOSA: context-aware adaptation for ubiquitous web access," Personal and Ubiquitous Computing, vol. 14, no. 4, pp. 301–320, Apr. 2009.

[Mar10]

Marcotte, Ethan (May 25, 2010). "Responsive web design". A List Apart. At: http://alistapart.com/article/responsive-web-design

[Men09]

Mendonça, H.; Lawson, L.; Vybornova, O.; Macq, B.; Vanderdonckt, J. A Fusion Framework for Multimodal Interactive Applications. In Proceedings of the ICMI-MLMI 2009, Cambridge, USA, 2009. pp 161 - 168

[Mit07]

Mitrovic, N., Royo, J. A. and Mena, E. (2007). Performance analysis of an adaptive user interface system based on mobile agents. In Proc. DSV-IS '07. New York: ACM Press, 29--44.

[Mit09]

Mitrovic, N. Royo, J. A. and Mena, E. (2009) Adaptive Graphical User Interfaces: An Approach Based on Mobile Agents, Networks, pp. 1-36, 2009.

[Mor12]

Morfeo Project, (2012) Context of Use Meta model. Available online at: http://forge.morfeo-project.org/wiki_en/index.php/Context_Of_Use_ Meta-model#Introduction

[Moh06]

Mohomed, I., Cai, J. C. , Chavoshi, S. and Lara, E. D. Context-Aware Interactive Content Adaptation, MobySys'06, pp. 42-55, 2006.

[Mot12]

Motti, V. (2011). Deliverable 2.1.1 – CARF and CADS (R1). Publically available online at: http://www.serenoa-fp7.eu/wp-content/uploads/2012/10/SERENOA_D2.1.1.pdf

[Mot12]

Motti, V. (2012). Deliverable 2.1.2 – CARF and CADS (R2). Publically available online at: http://files.morfeo-project.org/serenoa/public/deliverables/M18/SERENOA_D2.1.2.pdf

[Mot13]

Motti, V. G., Vanderdonckt, J. 2013. A Computational Framework for Context-aware Adaptation of User Interfaces. RCIS'2013. In Proc. of the 7th IEEE International Conference on Research Challenges of Information Science.

[Mot13b]

Vivian Genaro Motti, Dave Raggett, Sascha Van Cauwelaert, and Jean Vanderdonckt. 2013. Simplifying the development of cross-platform web user interfaces by collaborative model-based design. In Proceedings of the 31st ACM international conference on Design of communication (SIGDOC '13). ACM, New York,

NY, USA, 55-64. DOI=10.1145/2507065.2507067 http://doi.acm.org/10.1145/2507065.2507067

[Mot13c]

Vivian Genaro Motti; Ugo Sangiorgi and Jean Vanderdonckt. 2013. Enhancing Collaborative Sketching Activities with Context-aware Adaptation Guidelines. In Proceedings of the 19th Brazilian Symposium on Multimedia and the Web. (WebMedia 2013) ACM, New York, NY, USA.

N

[Neb11]

Nebeling, M., Matulic, F. and Norrie, M. C. 2011. Metrics for the evaluation of news site content layout in large-screen contexts. In Proceedings of the 2011 annual conference on Human factors in computing systems (CHI '11). ACM, New York, NY, USA, 1511-1520. DOI=10.1145/1978942.1979164 http://doi.acm.org/10.1145/1978942.1979164

[Neb11b]

Nebeling, M. Context-Aware and Adaptive Web Applications: A Crowdsourcing Approach, In Proc of ICWE, 2011.

[Neg12]

Negulescu, M., Ruiz, J., Yang, L., and Lank, E.. 2012. Tap, Swipe, or Move: Attentional Demands for Distracted Smartphone Input. In Proceedings of the International Working Conference on Advanced Visual Interfaces (AVI '12), Genny Tortora, Stefano Levialdi, and Maurizio Tucci (Eds.). ACM, New York, NY, USA, 173-180. DOI=10.1145/2254556.2254589 http://doi.acm.org/10.1145/2254556. 2254589

[Nge07]

Ngeow, Y.C., Mustapha, A.K., Goh, E., Low, H.K.: Context-aware Workflow Management Engine for Networked Devices. International Journal of Multimedia and Ubiquitous Engineering (IJMUE) 2(3), 33-47 (2007).

[Nic02]

Nichols, J., Myers, B. A., Higgins, M., Hughes, J., Harris, T. K., Rosenfeld, R., and Pignol, M.. Generating remote control interfaces for complex appliances. In CHI Letters: ACM Symposium on User Interface Software and Technology, UIST'02, Paris, France, 2002.

[Nig93]

Nigay, L. and Coutaz, J. "A Design Space for Multimodal Systems: Concurrent Processing and Data Fusion," InterCHI'93, p. 7, 1993.

[Nor86]

Norman, D. A. (1986). Cognitive engineering. User centered system design, 31-61.

[Nor89]

Norcio, A. F. and Stanley, J. Adaptive human computer interfaces: A literature survey and perspective. In: Transactions on System, Man and Cybernectics 19 (1989), no. 2, 399-408.

O

[Oli03]

Oliveira, J.M.P. and Fernandes, C.T. "A framework for adaptive educational hypermedia system," In: Workshop on Apps, Products and Services of Web-based Support Systems, IEEE/WIC, Halifax. Proc, 2003.

[Oli11]

Oliveira, J., Guerreiro, T. Nicolau, H. Jorge, J. and Gonçalves, D. 2011. BrailleType: unleashing braille over touch screen mobile phones. In Proceedings of the 13th IFIP TC 13 international conference on Human-computer interaction - Volume Part I (INTERACT'11), Pedro Campos, Nuno Nunes, Nicholas Graham, Joaquim Jorge, and Philippe Palanque (Eds.), Vol. Part I. Springer-Verlag, Berlin, Heidelberg, 100-107.

[OMG00]

Available at: http://cgi.omg.org/news/pr97/umlprimer.html

[Ope12]

CTIC Open DDR. (2012) Available at: http://www.openddr.org/

P

[Pat99]

Paternò, F. and Mancini, C. Designing Web Interfaces Adaptable to Different Types of Use. In Proc. of the Workshop Museums and the Web. Accessible at: http://www.archimuse. com/ mw99/papers/paterno/paterno.html, 1999.

[Pat11]

Paternò, F. and Sisti, C.. 2011. Model-based customizable adaptation of web applications for vocal browsing. In Proceedings of the 29th ACM international conference on Design of communication (SIGDOC '11). ACM, New York, NY, USA, 83-90. DOI=10.1145/2038476.2038493 http://doi.acm.org/10.1145/2038476.2038493

[Per00]

Perkowitz M. and Etzioni, O. "Towards adaptive Web sites: Conceptual framework and case study," Artificial Intelligence, vol. 118, pp. 245–275, 2000.

[Pre09]

Preuveneers, D., Victor, K., Vanrompay, Y., Rigole, P., Pinheiro, M. K., and Berbers, Y. "Context-aware adaptation in an ecology of applications." Context-Aware Mobile and Ubiquitous Computing for Enhanced Usability: Adaptive Technologies and Applications (2009): 1-25.

Q

[Qui12]

Quintilian Bibliography. Available online at: http://www.phillwebb.net/history/ancient/Quintilian/Quintilian.htm

R

[Rev00]

Revilla, L. F. and Shipman, F. M. 2000. Adaptive medical information delivery combining user; task and situation models. In Proceedings of the 5th international conference on Intelligent user interfaces (IUI '00). ACM; New York; NY; USA; 94-97. DOI=10.1145/325737.325791 http://doi.acm.org/10.1145/325737.325791

[Rom04]

Romer, Kay, and Friedemann Mattern. "The design space of wireless sensor networks." Wireless Communications, IEEE 11.6 (2004): 54-61.

[Rou00]

Rousseau, F. G.-M.s, J. Antonio Lima, José Valdeni de Duda, Andrzej. User Adaptable Multimedia Presentations for the WWW. In: World Wide Web, Toronto. VIII World Wide Web. 1999.

[Rou08]

Rouillard, J. "Adaptation en contexte : contribution aux interfaces multimodales et multicanal," [Report]: Habilitation à Diriger des Recherches; Université des Sciences et Technologies de Lille, France, 2008.

S

[Sch94]

Schilit, B.N., Adams, N.I. and Want, R., 1994. Context-Aware Computing Applications, In Proc. of the Workshop on Mobile Computing Systems and Applications, Santa Cruz, CA, pp. 85-90.

[Sch12]

Schwarz, T., Hennecke, F., Lauber, F. and Reiterer, H. (2012) Perspective+detail: a visualization technique for vertically curved displays. In Proceedings of the International Working Conference on Advanced Visual Interfaces (AVI '12), Genny Tortora, Stefano Levialdi, and Maurizio Tucci (Eds.). ACM, New York, NY, USA, 485-488. DOI=10.1145/2254556.2254648 http://doi.acm.org/10.1145/2254556. 2254648

[Sch04]

Scholtz, J. and Consolvo, S. 2004. Toward a Framework for Evaluating Ubiquitous Computing Applications. IEEE Pervasive Computing 3, 2 (April 2004), 82-88. DOI=10.1109/MPRV.2004.1316826 http://dx.doi.org/10.1109/MPRV.2004.1316826

[Sch13]

Schramme, R. "Adaptation of Graphical User Interfaces o Multiple Platforms: model, approach and two case studies" (2013) Master Thesis – Université catholique de Louvain

[Sen13]

Sendín, M., López-Gil, J. M., & López-Jaquero, V. (2013). Validation of a Framework for Enriching Human–Computer–Human Interaction with Awareness in a Seamless Way. Interacting with Computers, iwt046.

[Smi02]

Smith, A. S. G. and Blandford, A. N. N. MLTutor : An Application of Machine Learning Algorithms for an Adaptive Web-based Information System, International Journal of Artificial Intelligence in Education, pp. 1-22, 2002.

[Sot]

Sottet, J. S., Ganneau, V., Calvary, G., Coutaz, J., Demeure, A., Favre, J. M. and Demumieux, R.. "Model-driven adaptation for plastic user interfaces." In Human-Computer Interaction–INTERACT 2007, pp. 397-410. Springer Berlin Heidelberg, 2007.

[Spa03]

Spallek, H. Adaptive hypermedia: a new paradigm for educational software., Advances in dental research, vol. 17, pp. 38-42, Dec. 2003.

[Ste95]

Stephanidis, C., Savidis, A., and Akoumianakis, D. (1995). Towards user interfaces for all. In Conference proceedings of second TIDE congress (pp. 167–170).

[Str04]

Strang, T. and Linnhoff-Popien, C. A Context Modeling Survey, Workshop on Advanced Con- text Modeling, Reasoning and Management, The Sixth International Conference on Ubiquitous Computing, 2004.

T

[Thi94]

Thies, M. A. Adaptive User Interfaces, 13th World Computer Congress 94, Volume 2, K. Brunnstein, E. and Raulbold, E. Elsevier Science (North Holland) 1994.

U

[UMO03]

User Model Ontology, 2003. Available online at: http://www.u2m.org/2003/02/UserModelOntology.daml

[Usi07]

UsiXML Consortium. UsiXML, a General Purpose XML Compliant User Interface Description Language, UsiXML V1.8, 23 February 2007.

Available at http://www.usixml.org/index.php?view=page&idpage=6

V

[Van05]

Vanderdonckt, J., Grolaux, D., Van Roy, P., Limbourg, Q., Macq, B., Michel, 92 B., A Design Space for Context-Sensitive User Interfaces, Proc. of ISCA 14th Int. Conf. on Intelligent and Adaptive Systems and Software Engineering IASSE,2005 (Toronto, July 20-22, 2005), 2005

[Van08]

Vanderdonckt, J., Coutaz, J., Calvary, G., Stanciulescu, A. Multimodality for plastic user interfaces: Models, methods, and principles. In D. Tzovaras (ed.), Multi-

modal user interfaces: Signals and communication technology. Lecture Notes in Electrical Engineering, 2008, Springer, Berlin, pp 61–84.

[Ver12]

Veritas project, (2012) available at: http://veritas-project.eu/

[Voo13]

Vooren, R. v. "User Interface Adaptation by User Feedback" (2013) Master Thesis – Université catholique de Louvain

[Vri04]

de Vrieze, Paul, Patrick van Bommel, and Theo van der Weide. A generic adaptivity model in adaptive hypermedia. In Adaptive Hypermedia and Adaptive Web-Based Systems, pp. 344-347. Springer Berlin Heidelberg, 2004.

Y

[Yha12]

Yharrassarry, R. "Conception d'IHM sur la TV, de la connaissance des usages a la réalisation d'un service" Afterwork FLUPA Soirée "Ergonomie et Télévision du futur. Paris, March 6, 2012. Access in : http://blocnotes.iergo.fr/articles/concevoir-pour-la-tv-presentation-flupa/

W

[W3C]

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

[W3C03]

W3C Multimodal Interaction Framework. Larson, J., Raman, T., and Raggett, D. (2003) W3C Note. URL: http://www.w3.org/TR/mmi-framework/

[W3C09]

José Manuel Cantera Fonseca; Rhys Lewis. Delivery Context Ontology. 16 June 2009. W3C Working Draft. (Work in progress.) URL: http://www.w3.org/TR/2009/WD-dcontology-20090616/

[W3C10]

W3C Mobile Web Application Best Practices. Connors, A. and Sullivan, B. (2010) W3C Recommendation. URL: http://www.w3.org/TR/mwabp/# dependent -properties

Z

[Zel98]

Zelkowitz, M.V., Wallace, D., 1998. Experimental models for validating computer technology. IEEE Comput. 31 (5), 23–31.

[Zim07]

Zimmermann, A. 2007. 'Context Management and Personalization: A Tool Suite for Context- and User-Aware Computing' Ph.D. thesis, Fakultät für Mathematik, Informatik und Naturwissenschaften der Rheinisch-Westfälischen

Glossary

This section provides the main definitions that are frequently used in this thesis:

- **Context-awareness:** consists in gathering and using any relevant information from the user, platform or environment in order to provide users an application that suits better to their needs.
- **Model-based Approach:** consists in iteratively generating the application, following different granularity levels for development, mainly four abstraction levels: *Task and Domain*, *Abstract*, *Concrete* and *Final*, according to Cameleon Reference Framework [Cal03].
- **Adaptation Process:** corresponds to the complete cycle of transforming one or more aspects of an application, it usually involves three cyclic phases: (i) context gathering, (ii) adaptation reasoning, and (iii) model generation.
- **Adaptation Technique:** consists of the main unit of the adaptation process, i.e., one transformation of a resource of an application based in one (or more) specific condition(s) of the context of use.
- **Adaptation Method:** is a composition of different adaptation techniques aiming to perform an adaptation process.
- **Adaptation Strategy:** refers to the way in which the results of the adaptation are presented to the end user, e.g., by animating the transition between the original and the adapted UI.
- **Adaptation Principle:** is a quality of an application that must be taken into account while performing adaptation, such as continuity.
- **Adaptation Approach:** refers to the way in which the adaptation process is implemented, concerning for instance the methodology adopted or the architectural structure.
- **Adaptation Rule:** associates the context information (condition) with the adaptation technique (action), being used to define an adaptation process.
- **Adaptation Models:** are abstractions that formalize adaptation concepts in order to support developers and designers in defining and implementing adaptation.
- **Adaptability:** occurs when the systems are adapted in a manual manner, i.e., the users themselves are in charge of triggering, initializing or deciding about the adaptation process.
- **Adaptivity:** is the property of adapting systems in an automatic manner.
- **Adaptive User Interface:** user interfaces that automatically change.
- **Adaptable User Interface:** user interfaces that enable users to perform adaptation.
- **Intelligent User Interface:** apply available knowledge about the task to control the interaction, aiming to increase the efficiency and usability [Eis00].
- **Model-View Controller:** consists in an architectural approach that separates the representation of the information from the user's interaction. The model represents data and business rules, controller mediates the input, converting it to commands for the model or the view. The view represents the data (independently of format).

- **Plasticity:** consists in accommodating the characteristics of the context in the system in a flexible way, by optimizing the use of all resources available.
- **Nomadic UI's:** are interfaces that 'follow' the users' activities across multiple devices and platforms, in a pervasive and ubiquitous manner.

Appendix A. CAMM Description

The Adapters, represented by red classes in Figure 19 and Figure 20, refers to the agent or the set of agents that is responsible for triggering or supporting the decisions for the adaptation phases, they are defined as follows:

Adapter is the agent or the set of agents that is responsible for triggering or supporting the decisions for each of the adaptation phases;

- Examples: the end user that customizes the interactive system;
- Attributes: id (the identifier of the adapter, a unique value), name (the name associated to the adapter), and priority (could be in a qualitative approach a value like low, medium, or high according to the priority associated to the adapter);
- Methods: get() and set() (generic functions used to retrieve the information about the adapters available in a given moment and to associate it to the attribute values, instantiating the adapter);
- Relationships: is_generalization_of one or a set of User, System, and Third-Party, and triggers an AdaptationProcess

System is the computational application (e.g. a function, a program or an API) that interacts directly with the system;

- Examples: a web service;
- Attributes: id (the identifier of the system, a unique value), name (the name associated to the system) and its description (a summary of its definition and goals);
- Methods:
- Relationships: specializes an Adapter

Third-Party is an external agent able to intervene in the adaptation process;

- Examples: an agent;
- Attributes: id (the identifier of the third-party, a unique value), name (the name associated to the third-party) and its description (a summary of its definition and goals);
- Methods:
- Relationships: specializes an Adapter

User is the end user that is interacting with a system in a given moment, a human user;

- Examples: John Doe is the end user interacting with the system, his description includes his personal information, impairments (cognitive, motor, visual, etc.), and preferences;
- Attributes: id (the identifier of the user, a unique value) and its description (a summary of its definition and goals);
- Methods:
- Relationships: specializes an Adapter and also can compose one or a set of Context

When several users are considered in the decision, the adaptation is classified as crowd sourced [Neb11], and a mixed approach occurs when a combination of agents collaborate to take the adaptation decisions [Hor99]. The user is part of both Adapter and

Context, first as the agent responsible for taking adaptation decision and then its description is also relevant to composed contextual information.

Context

The context, represented by green classes in CAMM, refers to all the information that characterizes the context of use, the interaction scenario and that can be relevant for defining and executing the adaptation. It is defined mainly in terms of:

Context is all the information that characterizes the context of use, the interaction scenario and that can be relevant for defining the adaptation lifecycle;

- Examples: the user John Doe interacting with a tablet PC in a train;
- Attributes: id (a unique identifier value for that context), and a priority value (if in a qualitative approach could be high, medium, low levels, this information is useful to solve potential conflicts between adaptation techniques)
- Methods: get() (to retrieve information about the context, coming for instance from sensors in the environment), set() (function to instantiate the values for the context, such values can be treated and processed beforehand if necessary, e.g. to convert units), isAvailable() (to check whether there is information to be retrieved), isDynamic() (to check whether the information varies along the time), isValid() (to check whether the information is still holding);
- Relationships: is_composed_by at least one but not necessarily all User, Platform, Environment, Application, Quality, Element, Property, and instantiates a Justification, an Event and a ContextInformation

User (see previous definition for further information)

Platform is the device or set of devices used to interact, and all their relevant characteristics as accessories available, connections, technologies supported;

- Examples: a tablet PC with Android, specification about the connections, ports, compatibility, drivers, etc.;
- Attributes: id (unique identifier associated to the platform) and its description (a brief summary of the devices available and their characteristics);
- Methods:
- Relationships: a set of Platforms can compose a set of Contexts;

Environment is the scenario in which the interaction takes place, defined for instance in terms of light level, noise level, stability level, location, etc.;

- Examples: a train, with medium noise, light and stability level;
- Attributes: id (unique identifier associated to a given environment) and a description (a brief summary of the devices available and their characteristics);
- Methods:
- Relationships: can compose one or several Contexts;

Application is the description of the interactive system and its domain, described by (domain and data) models, (functional and non-functional) requirements, task tree, etc.;

- Examples: a safety-critical system, a medical system;
- Attributes: id (unique identifier associated to the environment) and its description (a brief summary of the characteristics of the environment);
- Methods: can compose a Context and is_associated_with Resources (the components of the UI's of the interactive system);
- Relationships: composes *Context* and has *Resources*;

Quality is a qualitative value used to evaluate certain characteristics of the context information (e.g. its validity, availability, precision);

- Examples: the information has a high level of precision (adopting a qualitative approach for implementation);
- Attributes: name (associated name with the quality, e.g. precision) and level (associated to the degree in which the quality is provided);
- Methods:
- Relationships: can compose one or several Contexts;

Element is one specific object of the context (i.e. the name of a context information);

- Examples: user;
- Attributes: name (the name given to the properties of the contextual information) and description (a brief summary explaining the name of the element);
- Methods:
- Relationships: can compose one or several Contexts;

Property is one specific attribute that characterizes one element of the context;

- Examples: the birthdate of the user;
- Attributes: name (the name given to the properties of the contextual information) and description (a short explanation about the property of the element);
- Methods:
- Relationships: can compose one or several Contexts;

Adaptation Core

The adaptation core involves the design decisions taken based on the processing of the contextual information available. This core includes inference and reasoning on top of the context in order to select and prioritize adaptation rules and their respective actions, completing a set of activities and functions performed to adapt some element of the interactive system;

Adaptation Process is the set of steps necessary to perform the adaptation, i.e. an adaptation lifecycle;

- Examples: given a specific context, the UI elements change, and are presented in a certain approach to the end user;
- Attributes: id (a unique identifier associated to an adaptation process);

- Methods: start() (to begin the adaptation process), pause() (to temporarily terminate the process), and stop() (to terminate the process);
- Relationships: is_triggered_by an Adapter, is_composed_by one or more AdaptationRules;

Adaptation Rule is a formal association connecting the context with the adaptation techniques, specifying how the system dynamically adapts according to the context [Koc00]. It can be structure according to the JECA approach [Nge07], defined as follows;

- Examples: if the user is dyslexic, then change the font type of the text;
- Attributes: id (a unique identifier associated to an adaptation rule), name (a unique name that characterizes the rule)
- Methods: calculatePriority() (to calculate the weight of the rule, based on context information provided) and apply() (to execute the rule in a given application);
- Relationships: composes AdaptationProcess, is_composed_by Justification, Event, Condition, and Action, can be part_of one or several Policies, and is_composed_by one or more AdaptationRules;

Justification is a reason associated to the context, that provides a rationale, and aids to prioritize the adaptation and to justify with qualitative or quantitative information the selection of one specific action, it forms the reasoning context in which evaluation of the specific JECA rule to be performed [Nge07];

- Examples: there is no information available about the environment (then a default scenario must be considered);
- Attributes: id (a unique identifier for the justification), weight, priority, argument (associated values to support reasoning);
- Methods: check() (verifies whether there is a justification available for a given instance of the context);
- Relationships: composes an AdaptationRule and is_instantiated_by the Context;

Event is a specific status or change of status regarding the system, the user interaction or the context that supports specific actions;

- Examples: when the device is rotated;
- Attributes: id (a unique identifier to the event), name (a word or statement to qualitatively identify the event) and description (a brief description that characterizes the event);
- Methods: detect() (the event is listened and detected by the application);
- Relationships: composes an *AdaptationRule* and is_instantiated_by an instance of the Context;

Condition is an association between a given element and a given instance by means of an operator (e.g. equal, greater than) that enables comparison and evaluation (an enumeration named OperatorType provides possible instances for the operators);

Examples: the user visual impairment is color blindness;

Attributes: id (a unique identifier for the given condition);

Methods: evaluate() (function to check whether the condition is valid or not)

Relationships: composes one or several AdaptationRules and it aggregates Value, Operator (specified by the enumeration OperatorType) and ContextInformation;

Value is the actual value that comes from the context of use, can be processed if needed (e.g. treated, converted), and verified according to the rule specification;

- Examples: 50;
- Attributes: name (a name associated to the value);
- Methods: process() (function to refine the value if needed, e.g. convert to a given unit or treat the information as necessary);
- Relationships: is_aggregated_with a *Condition* and instantiates a Technique;

Operator is the operator that permits a comparison between values;

- Examples: equal, greater than, different;
- Attributes: Type (an enumeration of possible instances is provided including equal, notequal, and, or, lessthan, greaterthan, lessthanorequal, greaterthanorequal);
- Methods:
- Relationships: is_aggregated_with a Condition and is_related_to a Technique;

ContextInformation is an element of the context that can be retrieved and evaluated;

- Examples: the list of visual impairments associated to the given user;
- Attributes: id (a unique identifier associated to the contextual information of interest);
- Methods: retrieves() (function to associate a value with the element)
- Relationships: is_aggregated_with a *Condition*, is_related_to a *Technique* and is_instantiated_by the Context;

Action is a function that defines and activates the execution of the adaptation;

- Examples: change the font size to 12;
- Attributes: name (the name of the action);
- Methods: execute() (function that performs a given action) and cancel() (to interrupt the execution of the action);
- Relationships: composes one AdaptationRule, specializes a Technique, is_presented_by a specific PresentationMethod, modifies a given Model, and is_composed_by a Method, a Classifier, a Resource, and a Parameter;

Technique is an operation that specifies a change in the system or in one or more properties of the system in order to adapt it;

- Examples: increase the font size;
- Attributes: id (a unique identifier associated with the technique), name (a name that characterizes the technique), reference (sources that define the technique, authors),

description (a brief summary explaining it), rationale (steps needed to accomplish it), example (illustrative uses for the technique), context (context associated with it), advantages (qualities that are enhanced when applying it), disadvantages (qualities that are hindered while applying it), sample (a piece of code that implements it, e.g. an URL linking to web service that implements the technique), images (illustrative pictures of its application), observation (commentaries and notes about it), categories (tags to classify it);

- Methods:
- Relationships: is_associated_with Value, Operator, and ContextInformation and is generalized_by an Action;

Policy is an abstraction of a technique that governs it;

Examples: if the user has low vision, but also a screen augmenter (as assistive device), there is no sense in augmenting the font size;

Attributes: id (a unique identifier associated to the policy);

Methods:

Relationships: is_associated_with *AdaptationRules* and is_part_of *Strategy*;

Strategy is an abstraction of a policy that governs it based on inferences performed with several contextual information;

Examples: a combination of two or more given policies;

Attributes: id (a unique identifier associated to the strategy);

Methods: apply() (a function to activate a given strategy);

Relationships: is_associated_with one or more given *Policy*

Method is one specific function applied to change an element of the interactive system;

- Examples: re-size;
- Attributes: name (a given name associated with the method);
- Methods: execute() (function to apply a given method);
- Relationships: is_aggregated_with an *Action*

Classifier is a definition of amount (subset, union, intersection or complement);

- Examples: all, any;
- Attributes: type (a given name that characterizes the classifier);
- Methods: set() (a function to associate a classifier with a given action);
- Relationships: composes an Action;

Resource is a component of the UI or the system that can be subject to adaptation, different granularity levels are considered, e.g. navigation, UI images, tables, their rows, columns or cells;

- Examples: image;
- Attributes: type (the name of the given resource defining the UI element);

- Methods: set() (a function to associate a given resource with an action)
- Relationships: composes an Action;

Resource Property is a specific characteristic or attribute of a resource;

- Examples: width of a table;
- Attributes: name (a characterization of the resource property);
- Methods:
- Relationships: belongs_to *Resource*

Parameter is a value related to a given unit that specifies a parameter for the adaptation technique;

- Examples: +50%;
- Attributes: specification (a given value that characterizes the action);
- Methods: set() (a function to associate a given parameter to the action);
- Relationships: composes an Action;

Presentation Method is an explicit manner of presenting the adaptation to the end user aiming at avoiding disruption, possible types are listed as enumeration;

- Examples: an animation to present the re-sizing of an edit box;
- Attributes: id (a unique identifier associated with the presentation method), name (a short descriptive name associated with the presentation), and type (a characterization of the presentation);
- Methods: play() (a function to activate the execution of an presentation method), stop() (a function to stop the presentation method), pause() (a function to pause the presentation method), checkCompatibility() (a function to check whether the action is compatible with a given presentation method);
- Relationships: presents an Action;
- Enumeration: possible presentation types include animation, brighten, blind, bounce, clip, cross fade, collapse, dim, drop, expand, explode, fade, fade in, fade out, fold, highlight, morph, plug in, plug out, progressive rendering, puf, pulsate, scale, self healing, shake, size, slide, spotlight, transfer.

Model-based Approach

An abstract representation of the reality (of the system, its different perspectives and the UI) that by means of reification or specialization is transformed from one abstraction level to another:

Model is a formal definition of an interactive system, that can be decomposed in different abstraction levels, and complemented by different views, commonly expressed by means of a given notation (e.g. UML, XML, CTT);

- Examples: a UsiXML model specifying an interactive system;
- Attributes: id (a unique identifier of the model) and a description (the model definition);

- Methods: reify() (an specialization of a model to make it more concrete) and abstract() (transformation to a higher abstraction level);
- Relationships: is_composed_by one or several models of a *Task*, *AUI*, *CUI* and *FUI* and is_modified_by an Action;

Task is a set of actions and activities to be executed according to given constraints, as ordering, to achieve a specific goal while interacting with the system;

- Examples: an CTT or an HTA task tree;
- Attributes: id (a unique identifier associated to the task tree) and description (a definition of the task tree: nodes, relationships, properties, etc.);
- Methods: reify() (function to transform a task tree into an AUI model);
- Relationships: composes one or several Models;

AUI (Abstract User Interface) is the abstract definition of the system and its UI that is domain and platform-independent;

- Examples: a UsiXML model expressing an AUI model;
- Attributes: id (a unique identifier associated to the AUI model) and description (a definition of the AUI components);
- Methods: reify() (function to transform an AUI model into a more concrete definition, i.e. a CUI model) and abstract() (function to transform an AUI model into a more abstract definition, i.e. a Task Tree);
- Relationships: composes one or several Models;

CUI (Concrete User Interface) is a more concrete definition of the system, its UI and its components;

- Examples: a UsiXML model expressing a CUI model;
- Attributes: id (a unique identifier associated to the CUI model) and description (a definition of the CUI components);
- Methods: reify() (function to transform an CUI model into a more concrete definition, i.e. a FUI model) and abstract() (function to transform an CUI model into a more abstract definition, i.e. an AUI model);
- Relationships: composes one or several Models;

FUI (Final User Interface) is the (graphical) user interface to be presented and/or rendered to the end user;

- Examples: a running application;
- Attributes: id (a unique identifier associated to the FUI model) and description (a definition of the FUI components);
- Methods: abstract() (function to transform an FUI model into a more abstract definition, i.e. an CUI model);
- Relationships: composes one or several Models;

Appendix B. CAMM Schema

```
▼<xs:schema xmlns="http://CASFE2/Metamodel_CAMM" xmlns:xs="http://www.w3.org/2001/XMLSchema"
 targetNamespace="http://CASFE2/Metamodel_CAMM">
 ▼<xs:element name="Action">
  ▼<xs:complexType>
   ▼<xs:complexContent>
    ▼<xs:extension base="Technique">
     ▼<xs:all>
        <xs:element name="name" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
        <xs:element name="is_presented_by" type="PresentationMethod" minOccurs="1" maxOccurs="unbounded"></xs:element>
        <xs:element name="modifies" type="Model" minOccurs="1" maxOccurs="1"></xs:element>
      </xs:all>
     </xs:extension>
    </xs:complexContent>
   </xs:complexType>
  </xs:element>
 ▼<xs:complexType name="Technique">
  ▼<xs:all>
     <xs:element name="id" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="name" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="reference" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="description" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="rationale" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="example" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="context" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="advantages" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="disadvantages" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="sample" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="images" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="observation" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="categories" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="relates_to" type="Value" minOccurs="0" maxOccurs="1"></xs:element>
  </xs:all>
 </xs:complexType>
 ▼<xs:complexType name="Value">
  ▼<xs:all>
     <xs:element name="name" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
  </xs:all>
 </xs:complexType>
 ▼<xs:element name="Condition">
  ▼<xs:complexType>
   ▼<xs:all>
      <xs:element name="id" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
    </xs:all>
   </xs:complexType>
 </xs:element>
 ▼<xs:element name="AdaptationRule">
  ▼<xs:complexType>
   ▼<xs:all>
      <xs:element name="id" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
      <xs:element name="name" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
      <xs:element name="_..._" type="AdaptationProcess" minOccurs="1" maxOccurs="1"></xs:element>
      <xs:element name="is_part_of" type="Policy" minOccurs="0" maxOccurs="unbounded"></xs:element>
    </xs:all>
   </xs:complexType>
 </xs:element>
 ▼<xs:complexType name="AdaptationProcess">
  ▼<xs:all>
     <xs:element name="id" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="triggers" type="Adapter" minOccurs="1" maxOccurs="unbounded"></xs:element>
     <xs:element ref="AdaptationRule" minOccurs="0" maxOccurs="1"></xs:element>
  </xs:all>
 </xs:complexType>
 ▼<xs:complexType name="Adapter">
  ▼<xs:all>
     <xs:element name="id" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="name" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="priority" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="triggers" type="AdaptationProcess" minOccurs="1" maxOccurs="unbounded"></xs:element>
  </xs:all>
 </xs:complexType>
 ▼<xs:element name="Justification">
  ▼<xs:complexType>
   ▼<xs:all>
      <xs:element name="id" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
      <xs:element name="weight" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
      <xs:element name="priority" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
      <xs:element name="argument" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
      <xs:element name="instantiates" type="Context" minOccurs="0" maxOccurs="1"></xs:element>
    </xs:all>
   </xs:complexType>
 </xs:element>
 ▼<xs:complexType name="Context">
  ▼<xs:all>
     <xs:element name="id" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="priority" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element ref="Justification" minOccurs="0" maxOccurs="1"></xs:element>
  </xs:all>
 </xs:complexType>
```

```
<xs:element name="Event">
 <xs:complexType>
  <xs:all>
   <xs:element name="id" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
   <xs:element name="name" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
   <xs:element name="description" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
   <xs:element name="instantiates" type="Context" minOccurs="0" maxOccurs="1"></xs:element>
  </xs:all>
 </xs:complexType>
</xs:element>
<xs:complexType name="ContextInformation">
 <xs:all>
  <xs:element name="id" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
  <xs:element name="instantiates" type="Context" minOccurs="1" maxOccurs="unbounded"></xs:element>
 </xs:all>
</xs:complexType>
<xs:element name="User">
 <xs:complexType>
  <xs:complexContent>
   <xs:extension base="Adapter">
    <xs:all>
     <xs:element name="id" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="description" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
    </xs:all>
   </xs:extension>
  </xs:complexContent>
 </xs:complexType>
</xs:element>
<xs:element name="Platform">
 <xs:complexType>
  <xs:all>
   <xs:element name="id" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
   <xs:element name="description" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
  </xs:all>
 </xs:complexType>
</xs:element>
<xs:element name="Environment">
 <xs:complexType>
  <xs:all>
   <xs:element name="id" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
   <xs:element name="description" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
  </xs:all>
 </xs:complexType>
</xs:element>
<xs:element name="Application">
 <xs:complexType>
  <xs:all>
   <xs:element name="id" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
   <xs:element name="description" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
   <xs:element ref="Resource" minOccurs="1" maxOccurs="unbounded"></xs:element>
  </xs:all>
 </xs:complexType>
</xs:element>
<xs:element name="Resource">
 <xs:complexType>
  <xs:all>
   <xs:element name="type" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
   <xs:element ref="Application" minOccurs="1" maxOccurs="unbounded"></xs:element>
  </xs:all>
 </xs:complexType>
</xs:element>
<xs:complexType name="ResourceProperty">
 <xs:all>
  <xs:element name="name" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
 </xs:all>
</xs:complexType>
<xs:element name="Quality">
 <xs:complexType>
  <xs:all>
   <xs:element name="name" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
   <xs:element name="level" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
  </xs:all>
 </xs:complexType>
</xs:element>
<xs:element name="Element">
 <xs:complexType>
  <xs:all>
   <xs:element name="name" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
   <xs:element name="description" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
  </xs:all>
 </xs:complexType>
</xs:element>
<xs:element name="Property">
 <xs:complexType>
  <xs:all>
   <xs:element name="name" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
   <xs:element name="description" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
  </xs:all>
 </xs:complexType>
</xs:element>
```

```
▼<xs:complexType name="Policy">
  ▼<xs:all>
     <xs:element name="id" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element ref="AdaptationRule" minOccurs="1" maxOccurs="unbounded"></xs:element>
   </xs:all>
 </xs:complexType>
▼<xs:complexType name="Strategy">
  ▼<xs:all>
     <xs:element name="id" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="is_part_of" type="Policy" minOccurs="1" maxOccurs="unbounded"></xs:element>
   </xs:all>
 </xs:complexType>
▼<xs:complexType name="Operator">
  ▼<xs:all>
     <xs:element name="type" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
   </xs:all>
 </xs:complexType>
▼<xs:complexType name="PresentationMethod">
  ▼<xs:all>
     <xs:element name="id" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="name" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="type" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element ref="Action" minOccurs="1" maxOccurs="1"></xs:element>
   </xs:all>
 </xs:complexType>
▼<xs:complexType name="Model">
  ▼<xs:all>
     <xs:element name="id" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="description" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element ref="Action" minOccurs="0" maxOccurs="unbounded"></xs:element>
   </xs:all>
 </xs:complexType>
▼<xs:element name="Tasks">
  ▼<xs:complexType>
    ▼<xs:all>
       <xs:element name="id" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
       <xs:element name="description" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     </xs:all>
   </xs:complexType>
 </xs:element>
▼<xs:element name="AUI">
  ▼<xs:complexType>
    ▼<xs:all>
       <xs:element name="id" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
       <xs:element name="description" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     </xs:all>
   </xs:complexType>
 </xs:element>
▼<xs:element name="CUI">
  ▼<xs:complexType>
    ▼<xs:all>
       <xs:element name="id" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
       <xs:element name="description" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     </xs:all>
   </xs:complexType>
 </xs:element>
▼<xs:element name="FUI">
  ▼<xs:complexType>
    ▼<xs:all>
       <xs:element name="id" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
       <xs:element name="description" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     </xs:all>
   </xs:complexType>
 </xs:element>
▼<xs:element name="Method">
  ▼<xs:complexType>
    ▼<xs:all>
       <xs:element name="name" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     </xs:all>
   </xs:complexType>
 </xs:element>
▼<xs:element name="Parameter">
  ▼<xs:complexType>
    ▼<xs:all>
       <xs:element name="specification" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     </xs:all>
   </xs:complexType>
 </xs:element>
▼<xs:element name="Classifier">
  ▼<xs:complexType>
    ▼<xs:all>
       <xs:element name="type" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     </xs:all>
   </xs:complexType>
 </xs:element>
```

```
▼<xs:complexType name="System">
  ▼<xs:complexContent>
    ▼<xs:extension base="Adapter">
      ▼<xs:all>
         <xs:element name="id" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
         <xs:element name="name" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
         <xs:element name="description" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
       </xs:all>
     </xs:extension>
   </xs:complexContent>
 </xs:complexType>
▼<xs:complexType name="Third_Party">
  ▼<xs:complexContent>
    ▼<xs:extension base="Adapter">
      ▼<xs:all>
         <xs:element name="id" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
         <xs:element name="name" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
         <xs:element name="description" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
       </xs:all>
     </xs:extension>
   </xs:complexContent>
 </xs:complexType>
▼<xs:complexType name="OperatorType">
  ▼<xs:all>
     <xs:element name="equal" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="notequal" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="and" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="or" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="lessthan" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="greaterthan" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="lessthanorequal" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="greaterthanorequal" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
   </xs:all>
 </xs:complexType>
▼<xs:complexType name="ClassifierType">
  ▼<xs:all>
     <xs:element name="all" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="subset" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
   </xs:all>
 </xs:complexType>
▼<xs:complexType name="PresentationType">
  ▼<xs:all>
     <xs:element name="animation" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="brighten" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="blind" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="bounce" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="clip" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="cross_fade" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="collapse" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="dim" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="drop" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="expand" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="explode" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="fade" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="fade_in" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="fade_out" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="fold" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="highlight" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="morph" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="plug_in" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="plug_out" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="progressive_rendering" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="puf" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="pulsate" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="scale" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="self_healing" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="shake" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="size" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="slide" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="spotlight" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
     <xs:element name="transfer" type="xs:string" minOccurs="0" maxOccurs="1"></xs:element>
   </xs:all>
 </xs:complexType>
</xs:schema>
```

Appendix C. CARF Instances

What	Why[18]	How[19]	To What[20]	Who	When	Where
Navigation	Functionality	Adapt Navigation	Environment	Developer	Design Time	Client
Presentation	Maintainability	Adapt Presentation	Platform	End-user	Compilation Time	Proxy
Content	Portability	Adapt Content	User	System	Run Time	Server
Animation	Reliability			Third-Party		
Audio	Usability					
Graphic						
Image						
Text						
UI Element						
Video						

[18] Appendix D (based on [ISO9126])

[19] Appendix E

[20] Appendixes F, G and H

Appendix D. Software Qualities [ISO9126]

Functionality	Maintainability	Portability	Reliability	Usability
Accuracy	Analyzability	Adaptability	Compliance	Attractiveness
Compliance	Modifiability	Co-existence	Fault Tolerance	Compliance
Interoperability	Compliance	Conformance	Maturity	Learnability
Performance	Stability	Compliance	Recoverability	Operability
Security	Testability	Instalability		Understandability
Access Control		Replacing Capability		Effectiveness
Authorization				Efficiency
Authentication				Time
Accountability				Resources
Availability				User Satisfaction
Confidentiality				
Integrity				
Privacy				
Suitability				

Appendix E. Adaptation Techniques (154)

A detailed description and also further references about each of the adaptation techniques listed below is available at http://sites.uclouvain.be/mbui/caa/adaptation_techniques.pdf.

Navigation (25)	Presentation (23)	Content (13)	Audio (11)	Image (30)	Text (30)	UI Element (11)	Video (11)
Accessible Navigation	Alternative View	Change Direction	Change Bit Rate	Adjust Shape	Additional Explanation	Change Size	Change Frame Resolution
Annotate Links	Attach	Change Modality	Change Sample Rate	Collapse to zoom	Align	Replace	Change Frame Spatial Quality
Organize Alphabetically	Augment	Conditional Text	Convert Channel	Change Brightness	Alter Fragments	Adapt Form	Skip Frame
Navigate Alternatively	Collapse to Zoom	Frame Based	Convert Format	Change Color Balance	Auto-correct	Adjust Form	Semantic Summary
Organize Chronologically	Detach	Filter	Translate Language	Change Color Map	Change Background	Expand TextBox	Remove Shot
Concern-Sensitive Navigation	Distribute	Segment by Index	Translate Modality	Translate Color	Change Color	Split Table	Transcode Format
Categorize	Fish eye	Personalize	Change Volume	Change Contrast	Explain Comparatively	Transform Table	Change Speed
Guide Directly	Full-screen	Re-size	Change Speed	Change Quantization	Compare	Adaptive Menu	Select
Disable Link	Large Screen	Re-order	Truncate	Change Transparency	Change Contrast	Split Interface	Reduce
Elise First Sentence	Migrate	Screenfinity	Simplify	Digital Compose	Describe	Moving Interface	Replace
Generate Links	Mirror	Stretch	Summarize	Matte	Dim Fragments	Visual Popout In-	Synthesize

						terface	
Hide Links	No-script	Suggest (Recommend)		Optical	Dislexie		
Organize Hierarchically (within a page)	Optimize	Translate to Vocal		Daltonize	Explain Variants		
Hierarchical Structure (within an application)	Overview + Detail			Demosaicing	Change Font Family		
Segment by Index	Perspective + Detail			Differentiate	Change Font Style		
Linear Structure	Print			Enlarge	Highlight		
Adapt Map	Re-distribute			Elise Image (Reduce)	Orient		
Mesh Structure	Re-mold			Bayer Filter	Outline		
Outline	Re-size			Hash Geometrically	Pre-requisite (Explain)		
Paginate	Replace			High Dynamic Range Imaging	Change Readability		
Organize by Popularity	Single Column			Register Image	Change Serif		
Remove Link	Expand Textbox			Morph Image	Similarity		
Search	Transcode			Recognize Image	Simplify		
Simplify Input Controls	Decompose			Segment Image	Re-Size		
Sort Links				Interpolate	Space		

Suggest Link				Reduce	Stretch Text		
				Rotate	Summarize		
				Scale	Sort		
				SmartView	Translate		
				WebThumb	Truncate		
				Binary Transcode	Zoom		

Appendix F. Environment

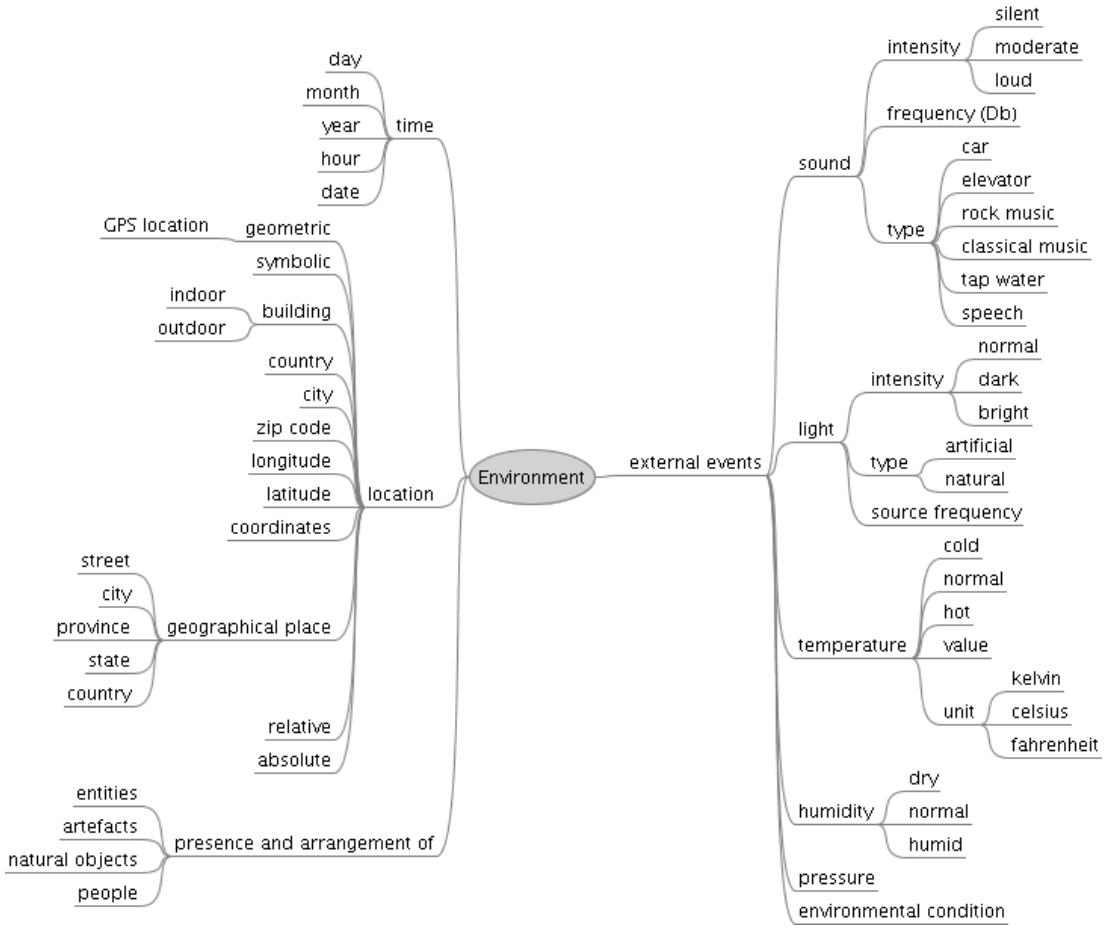


Figure 77. Context Information for CARF: Environment

Appendix G. Platform

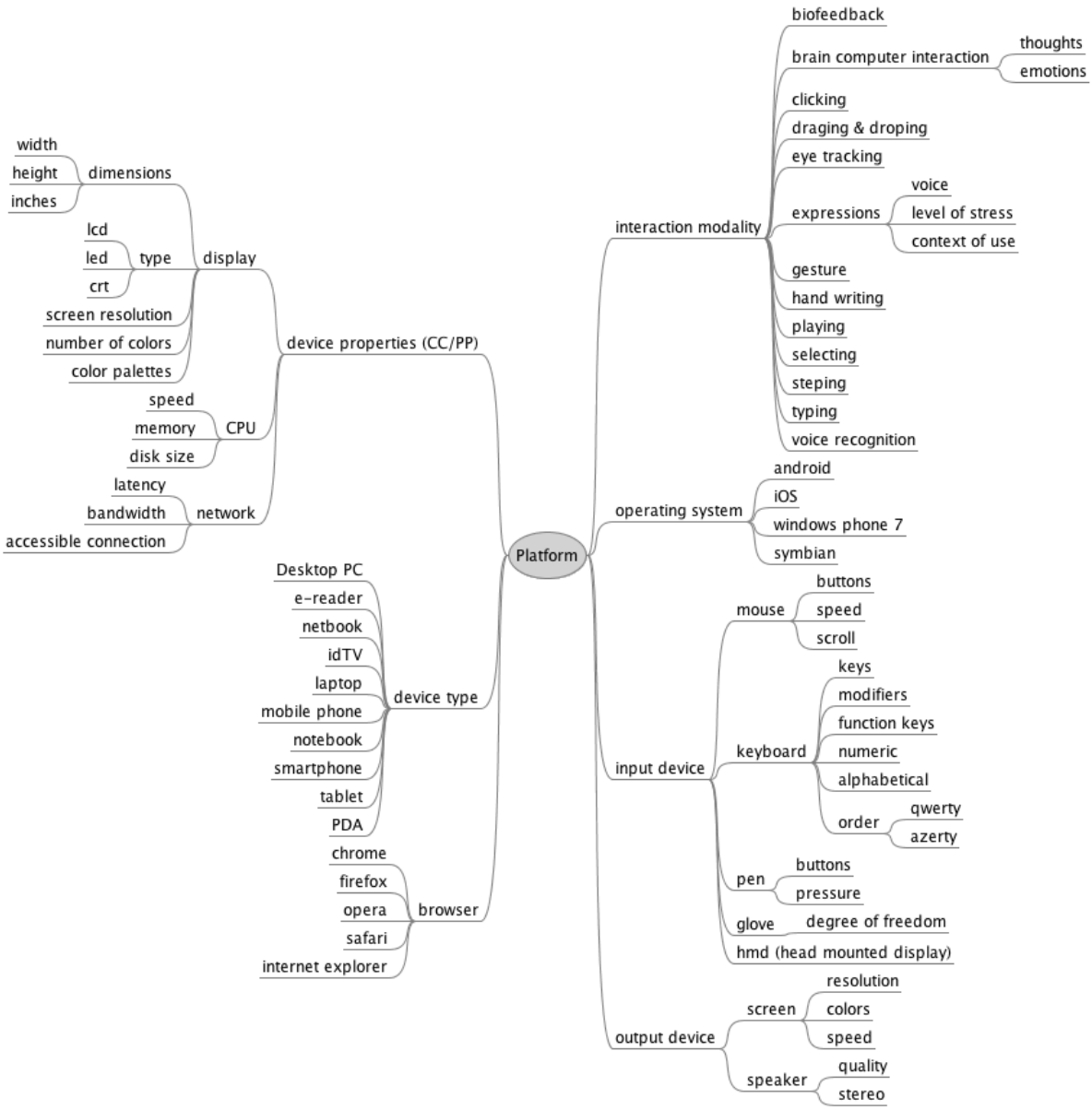


Figure 78. Context Information for CARF: Platform

Appendix H. User (I)

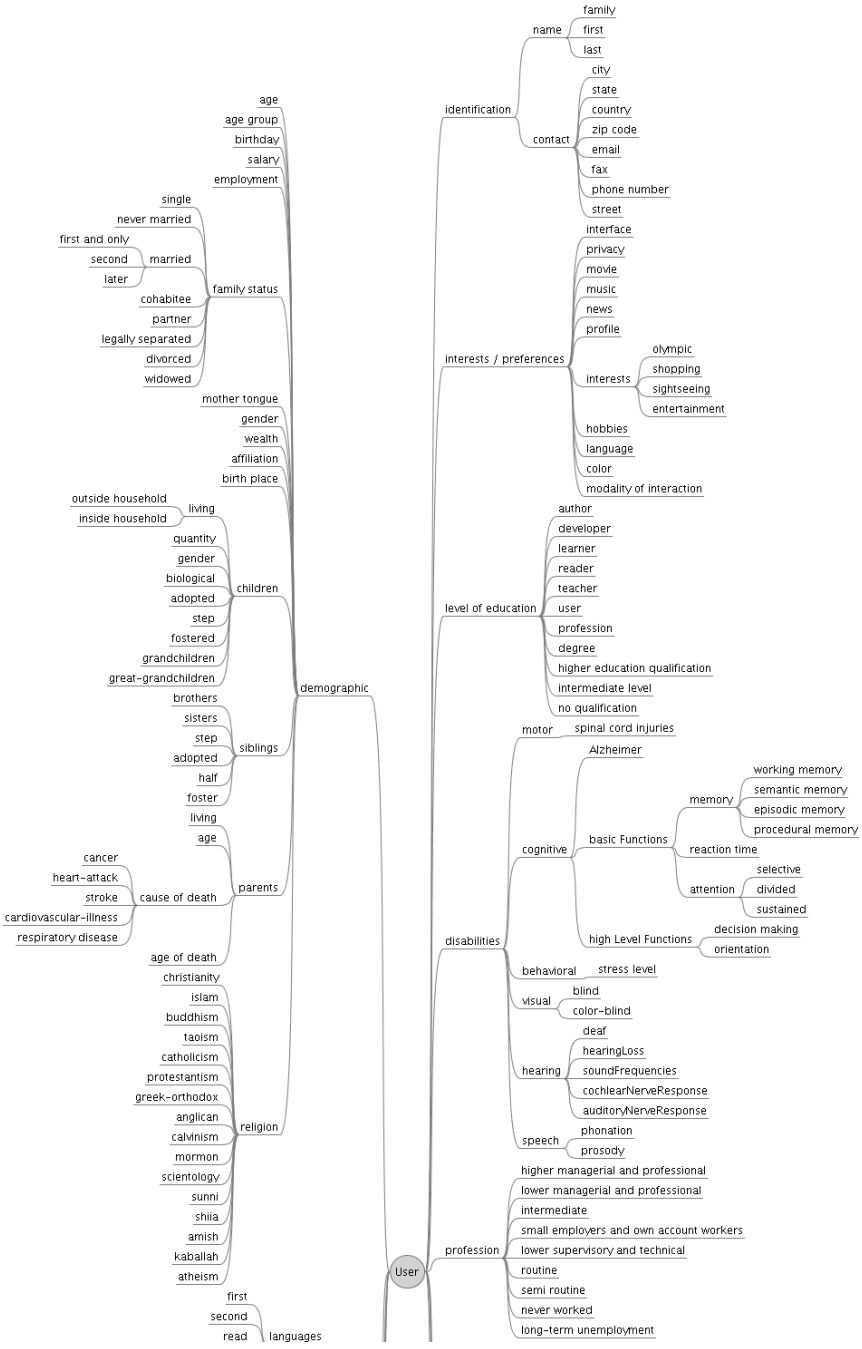


Figure 79. Context Information for the CARF: User (I)

Appendix I. User (II)

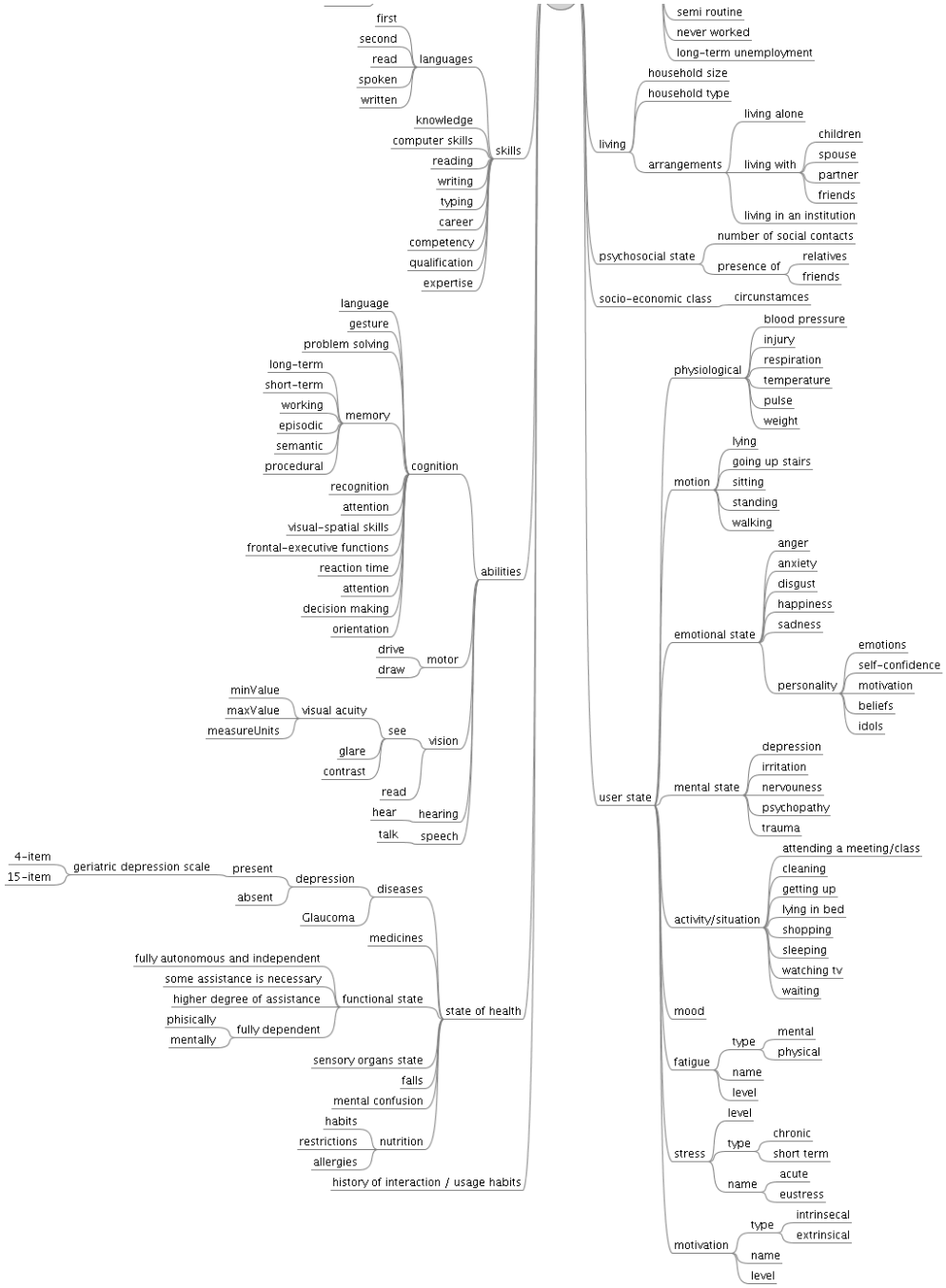


Figure 80. Context Information for the CARF: User (II)

Appendix J. Adaptation Meta-models

This appendix section presents a selection of images of related meta-models for adaptation:

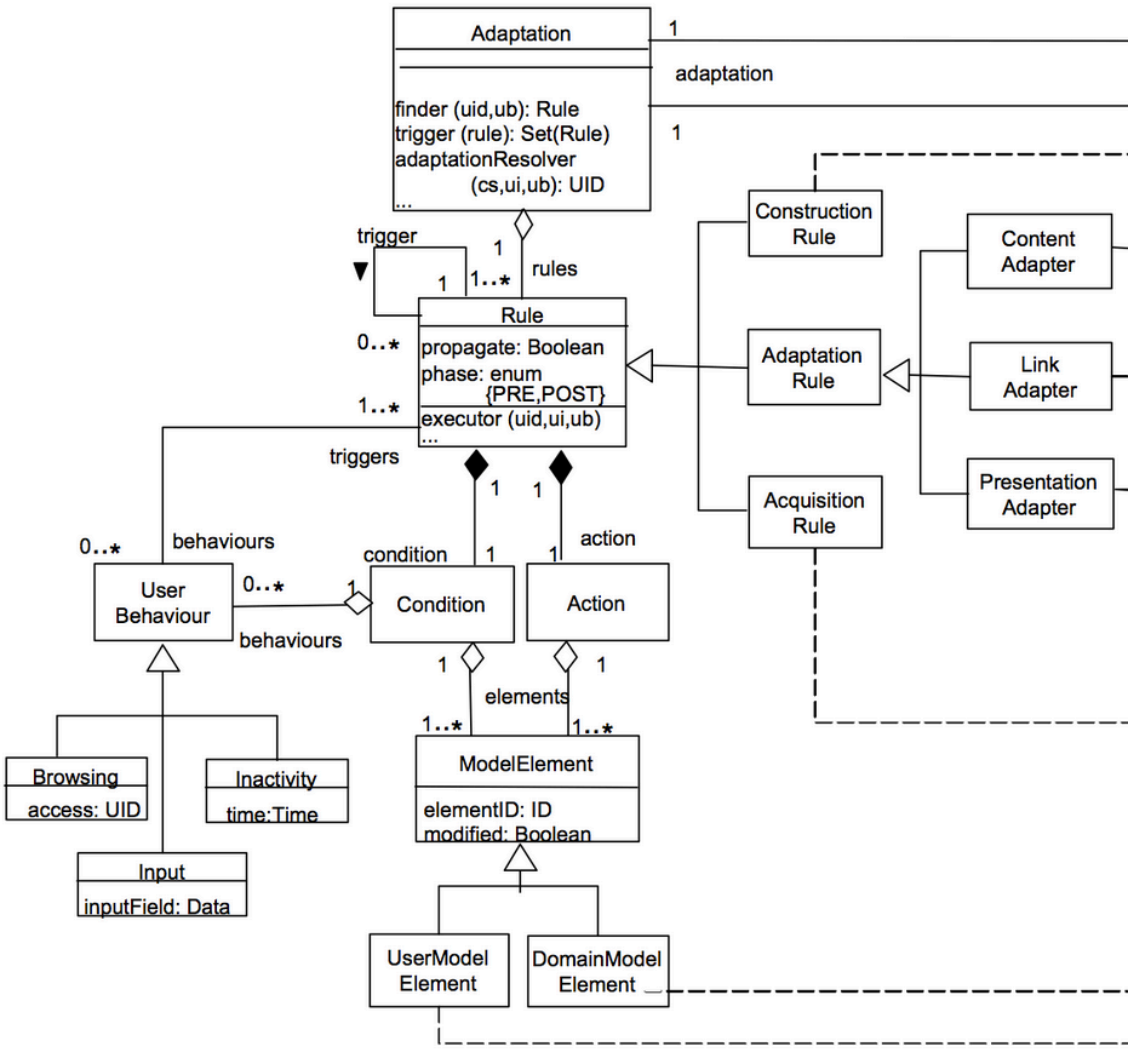


Figure 81. The Munich model (partial view) [Koc01]

The Munich Reference Model (Figure 81) covers adaptive hypermedia applications, including also a domain model and a user model. It covers mainly context (user) and rules (condition, actions) [Koc01].

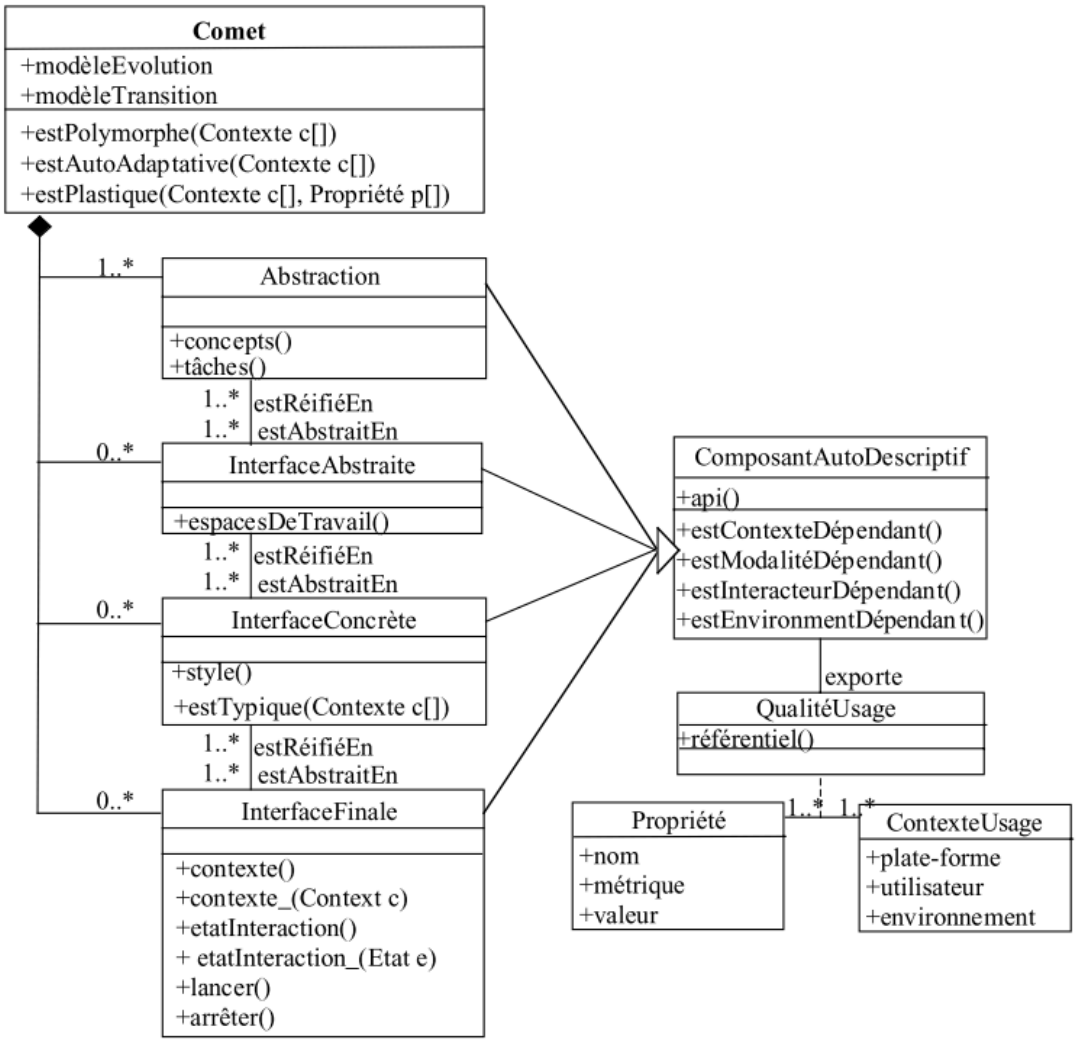


Figure 82. Comets [Cal05]

The Comets Model [Cal05] (Figure 82) targets at plasticity and adaptation according to the context of users, platform and environments. Comets refer to a new generation of widgets, which changes its format and shapes depending on their instantiations. Different abstraction levels are covered: tasks, abstract, concrete and final. This model covers mainly the widget definition (comet), the abstraction levels and the quality of the context.

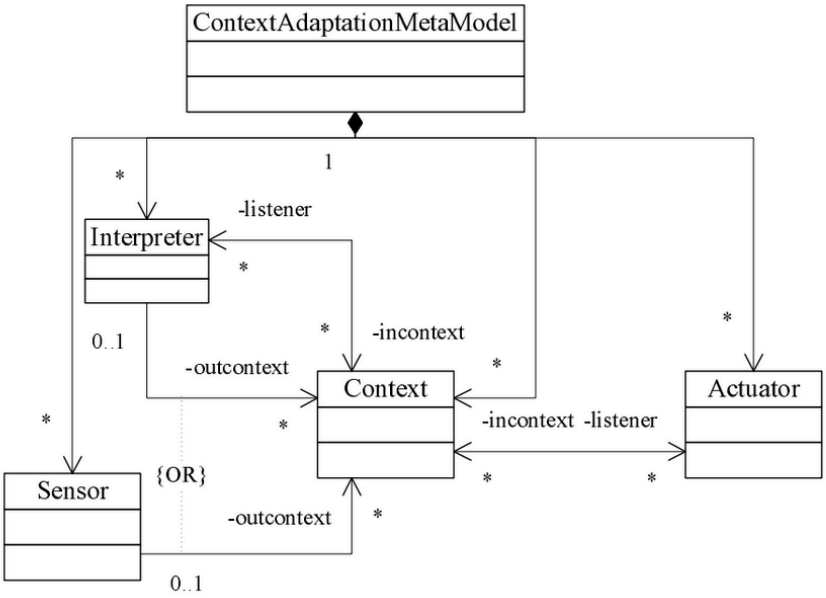


Figure 83. Meta-model of the K-Model [Fah05]

This meta-model (Figure 83) includes components for gathering the context (Sensors) to process it (Interpreter) and to react accordingly (Actuator) [Fah05].

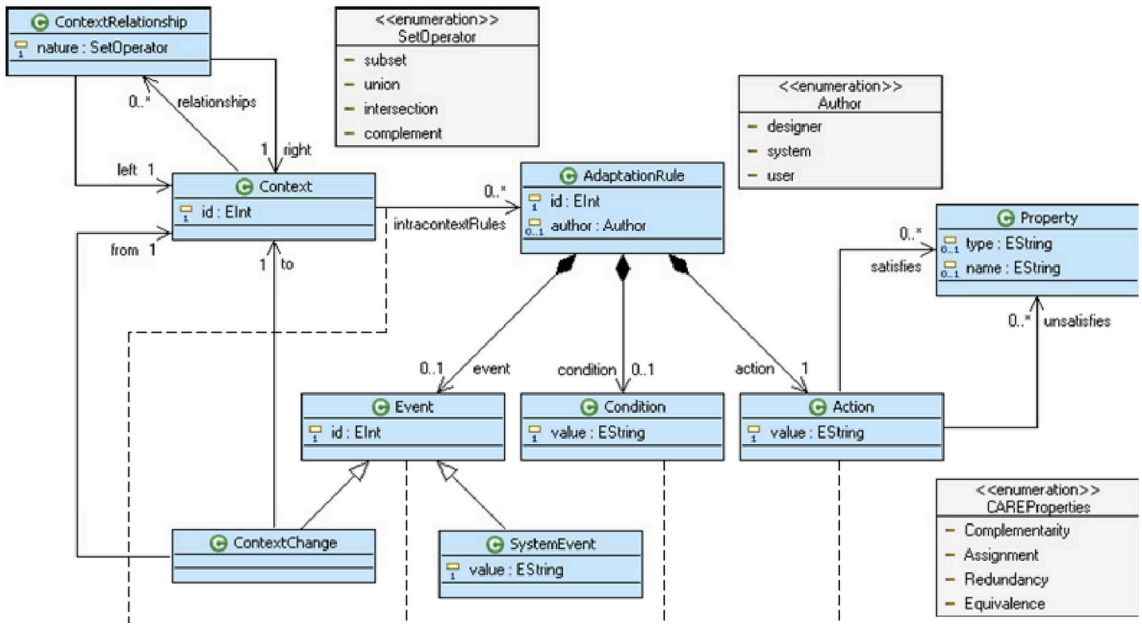


Figure 84. Meta-model for Adaptation Rules [Gan07]

This meta-model (Figure 84) covers the adaptation rules aiming also the plasticity of the UI. It covers the context information, and rules associating: events, condition and actions [Gan07].

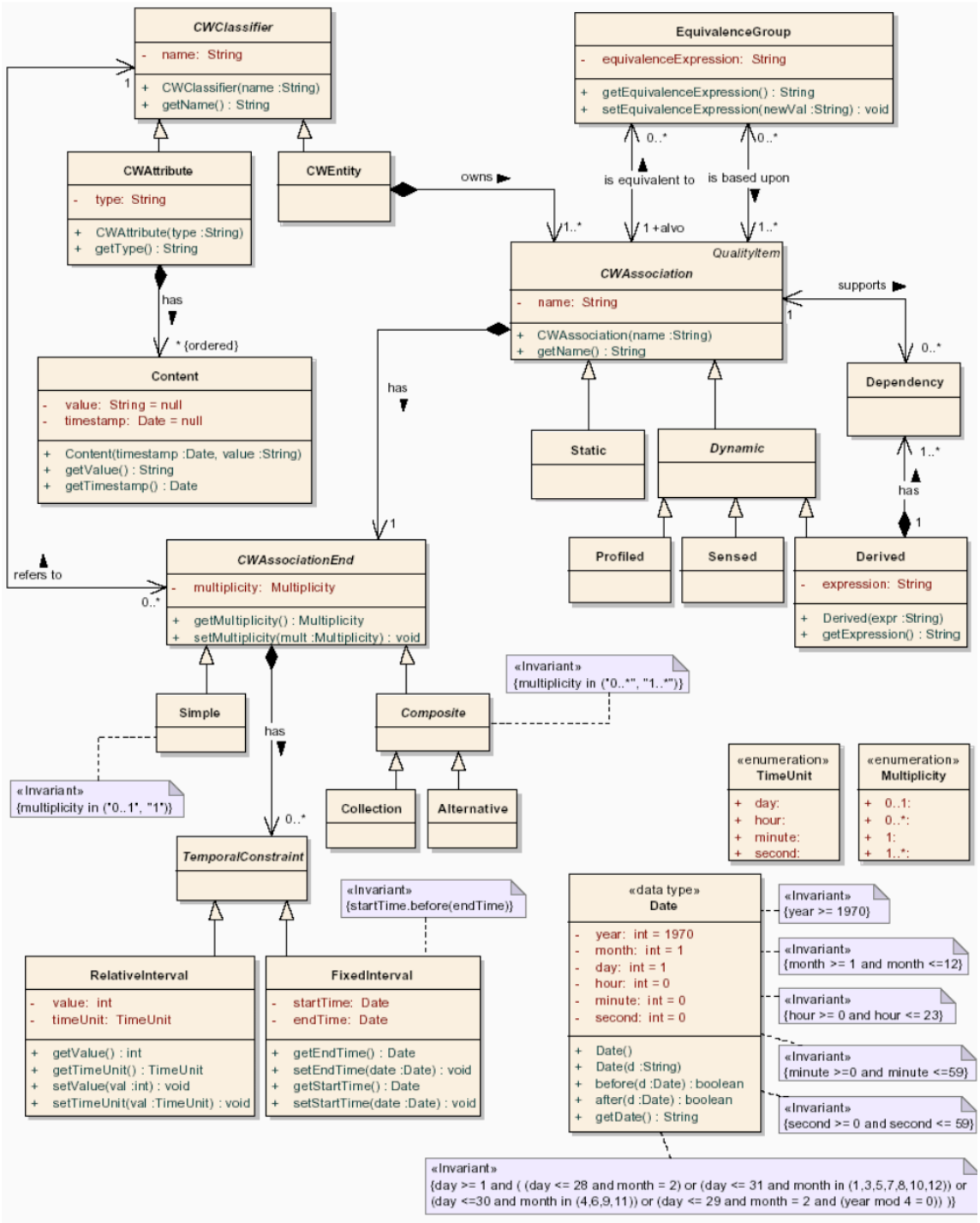


Figure 85. A MOF model for context-aware mobile applications [Far07]

The MOF-meta-model of [Far07] covers mainly the context information and the adaptation rules for the adaptation (Figure 85).

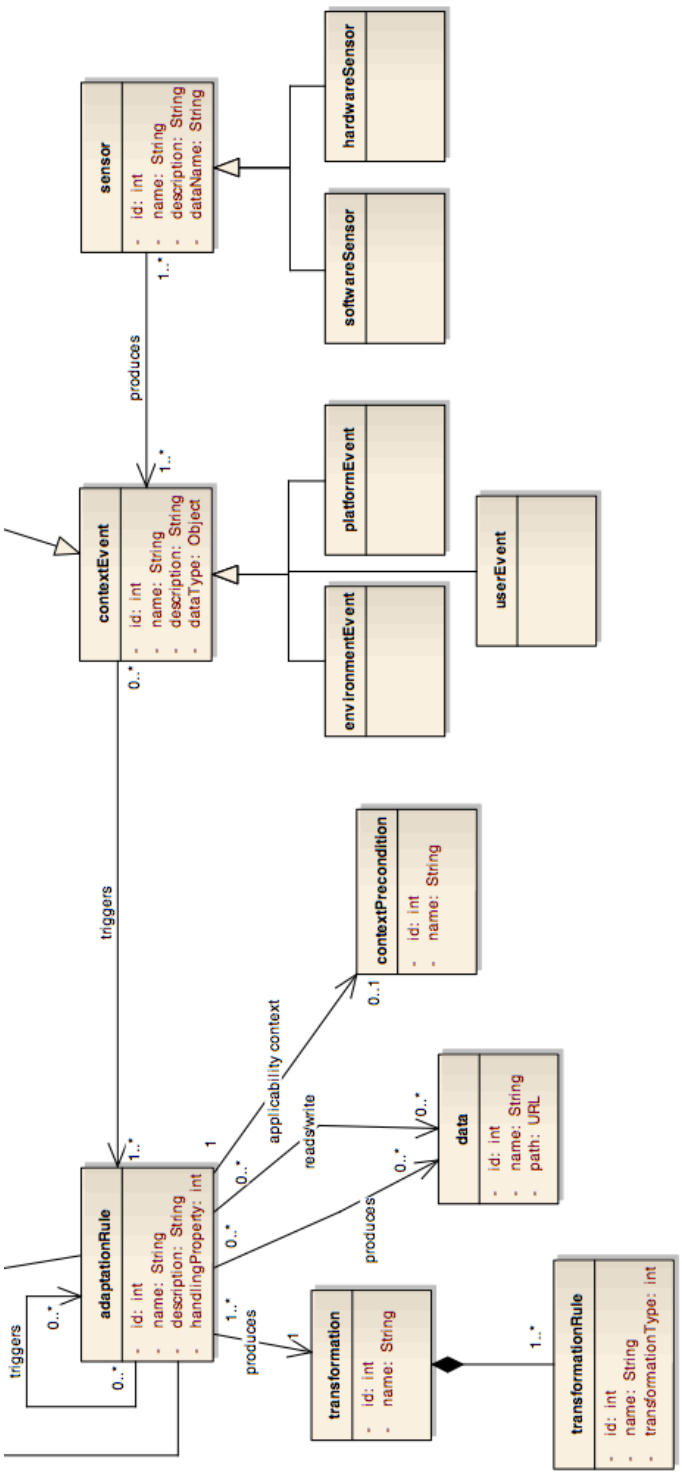


Figure 86. An adaptation rules meta-model (partial view) [Lóp09], [Lóp10]

The meta-model of [Lóp10] mainly covers the adaptation rules. The components of the model include: pre conditions, events, transformations, data and sensors (Figure 86).

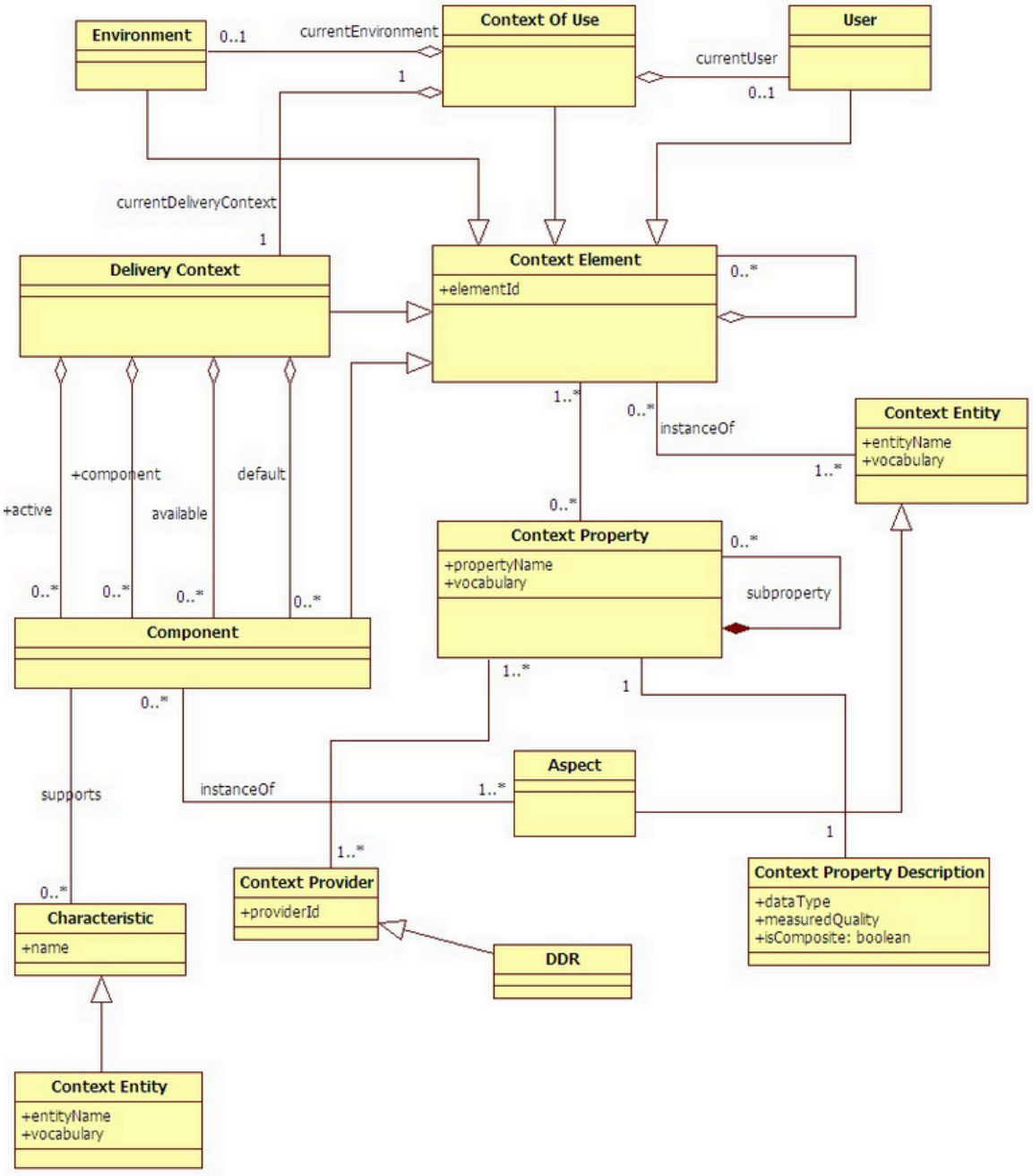


Figure 87. The MORFEU meta-model for context of use [Mor12]

The meta-model diagram (Figure 87) implemented for the MORFEU project covers mainly the context of use. It includes concepts as: the user, the elements, the properties, the entities, the aspects and the descriptions [Mor12].

Appendix K. Feature Table

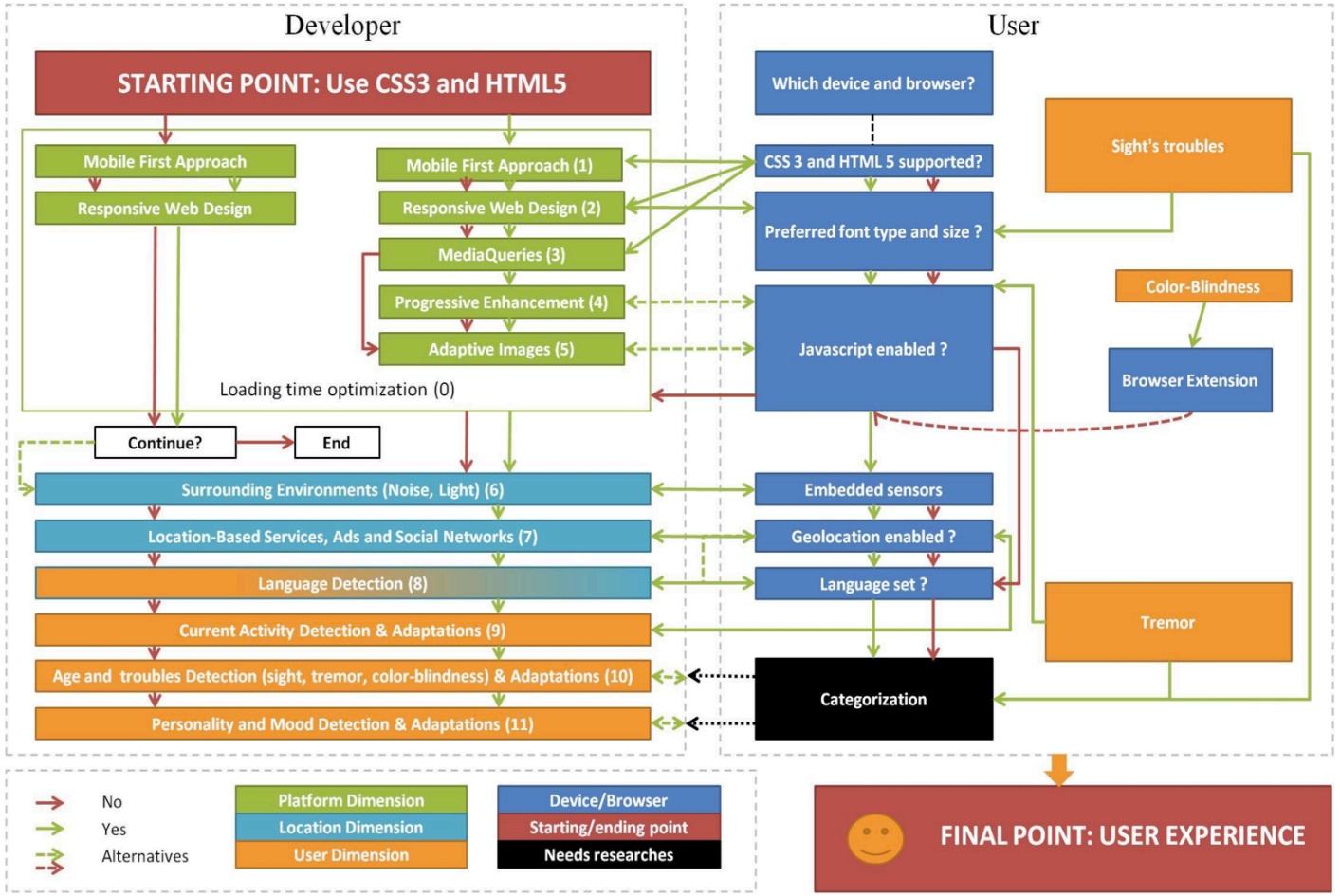


Figure 88. Adaptation decision table [Car13]

Appendix L. Design, Test and Evaluation

For decades, different adaptations of user interface have been presented and discussed regarding their pros and cons on the task performance and the user satisfaction. Little attention, however, has been directed to isolate and understand which aspects of adaptation them more successful [Gaj06].

- **Policies and Evaluation Metrics.** Mohomed *et al.* (2006) defined prediction policies based on statistical measures to execute always the most frequently requested adaptations. In this scenario, they defined metrics of evaluation based in the percentage of errors in predicting the desired layout for the users. In their case studies the user profiles and platform types were used to adapt image properties (as size and resolution). They state that specific context information has a different impact on adaptation and that grouping users into communities seems to significantly improve the prediction's accuracy.
- **Usability Challenges.** Jameson (2003) classified as the five most significant usability challenges for adaptive interfaces: predictability and transparency, controllability, unobtrusiveness, privacy, and breadth of experience. His work focuses on usability and adaptive interfaces, or systems that learn from the user's behavior and react accordingly [Sch04].
- **Costs and Benefits.** Gajós *et al.* (2006) observed that it is important to analyze the costs in current from adaptations. They identified properties that are likely to impact the adaptation benefits, such as: the ease of use, discoverability and predictability, mental and physical demand, performance, satisfaction, frustration, control, efficiency and confusion (due to the adaptation).
- **Evaluation Criteria.** Gajós *et al.* (2008) also analyzed the impact of predictability and accuracy for adaptive users interfaces. They state that adaptation may optimize the interaction according to the users' style and task, however the inherent unpredictability of adaptive UI's may disorient users causing more harm than good. To analyze the satisfaction of adaptation for end users, they defined as relevant criteria: usefulness, predictability, and the levels of knowledge (understanding), frustration, confusion, satisfaction, control and efficiency. In this work they concluded that, although both predictability and accuracy affect users satisfaction, accuracy has a higher impact on performance of adaptive UI's.
- **E-frame.** Arhippainen (2009) proposed a framework to evaluate the user experience, it is method-independent and it can be used for planning and conducting tests. She states that the best practice to study user experience is to use several methods together; she also proposed ten heuristics to aid the design and evaluation of user experience. The criteria are presented in Figure 89.
- **Adaptability vs. Adaptivity.** Frias-Martinez *et al.* (2009) states that giving the control to the users can reduce the effect of incorrect adaptation. However, the cost of the increased controllability is the additional effort required from the users. They investigated

the differences between adaptability and adaptivity for end users. They noticed that for a digital library scenario, users not only performed better in the adaptive version, but also they perceived more positively to the adaptive version. Moreover, they noticed that users cognitive styles have a great impact on how users perceive adaptation.

- **Design Patterns and Usability Principles.** Motivated by the fact that mobile interfaces can improve the users' experience with the surrounding environment, Ginige *et al.* (2012) investigated common usability issues in the representation of spatial data in mobile application. This investigation led to the definition of two design patterns, to be adopted as usability principles that a well-designed user interface should adopt. The design pattern named *infinitive area+ context* proposes the definition of colored areas with visual metaphors that provide information about the contents hidden in the off-screen space. The colors and dimensions of the metaphors can correspond to qualitative or quantitative information. The pattern of *infinitive area + context* was extended with multimodality by associating the visual metaphors with auditory and tactile feedback.
- **Evaluation Framework for Customization.** categorizes customization into different dimensions. The benefit of this evaluation framework is threefold. First, it allows a structured, uniform view to better understand the various aspects of customization. Second, it can be used as a conceptual framework for evaluating existing approaches on customization. Third, it may be employed for developing next generation customization approaches. Basically, this evaluation framework comprises two orthogonal dimensions context and adaptation and the mapping in between represented by the notion of customization (Figure 90) [Kap03].
- **Evaluation Framework for Ubiquitous Applications.** Scholtz and Consolvo (2004) believe that the lack of a widely accepted framework for user evaluations of ubiquitous applications hampers their efforts. A framework can help researchers to compare results, to create design guidelines, to develop evaluation techniques, to understand the appropriateness of different evaluation techniques, and to develop a more complete structure so they avoid overlooking key areas of evaluation. They defined 9 areas and 34 metrics for evaluating ubiquitous applications (Table 14). Although targeted at ubiquitous computing, we believe these criteria are also relevant for CAA.

Given that many influencing factors are involved, evaluating applications is a complex and challenging task. Moreover, biased conclusions can easily rise. The evaluation tasks become even more difficult if the system is adaptive [Hoo97]. The evaluation of adaptive systems is particularly complex as the results of the adaptation are personal to a specific user's set of circumstances and, thus, an empirical study is the most appropriate strategy for evaluation [Smi02]. Although adaptive research has produced promising results (in terms of providing personalized information) [Bru96], a weakness of this research field is the lack of comprehensive empirical studies to measure the usefulness of adaptation within and between the systems. One reason for this is that there is no standard or agreed evaluation framework to measure the effectiveness of adaptation.

One typical approach to determine the effectiveness of adaptation has been to compare the performance of an adaptive system against a version of the system with adaptation disabled. Adaptivity should preferably be an inherent part of a system [Hoo97], and so if it is removed from the system, the system may not be fully functional [Smi02].

Another approach consists in enabling users to manually modify the adaptation and then provide a feedback to the system (evaluation based on users' perspective).

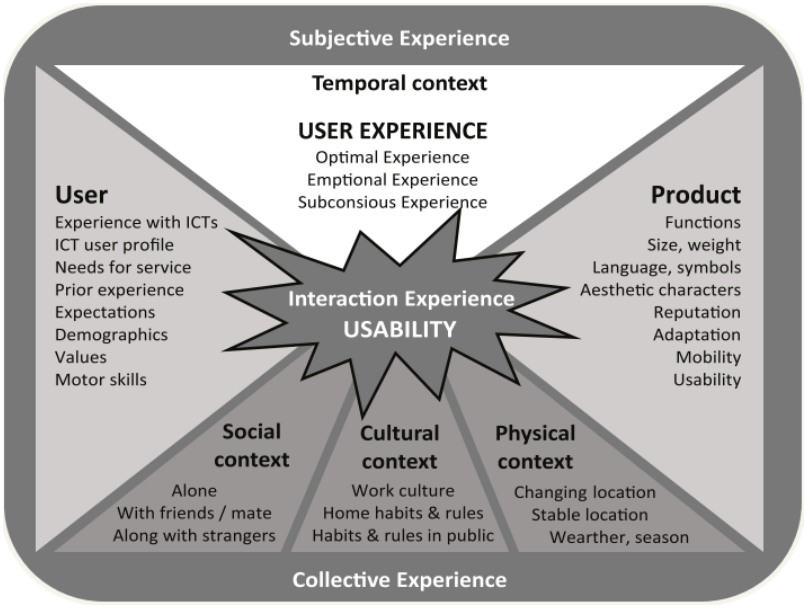


Figure 89. Framework for evaluation of the user experience [Arh09]

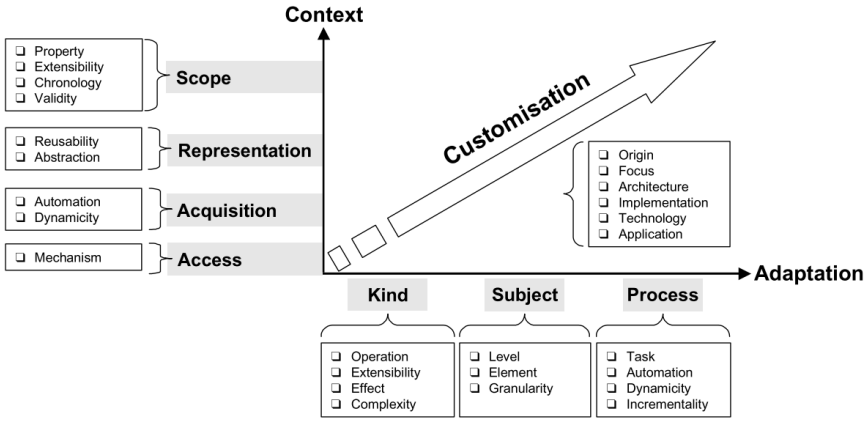


Figure 90. Evaluation framework for customization [Kap03]

Jameson (2003) defined usability criteria and principles that are closely involved in the evaluation of adaptation. He also identified potential threats and their respective countermeasures.

One approach to evaluate adaptive systems includes measuring the usage errors and slips of the user. While the former consists in users having the wrong intentions while executing a task (i.e., a misunderstanding of the UI), the later occurs when users had the correct intentions but executed the task in a wrong way (e.g., inadequate UI's) [Bre09]. The authors believe that slips can be prevented by adapting the UI according to the user, and errors are prevented by adhering to design heuristics.

Adaptivity is not equally beneficial under all conditions [Lav10], and given that AUI's are used in complex and dynamic environments, the context variables must be taken into account. The user preferences vary, e.g., according to the frequency of the tasks, users' age, task complexity level, and the user involvement. As such, variable levels of adaptivity are more beneficial than just one level. Even though, the following aspects must be careful-

ly considered too: (i) the frequency of the tasks, (ii) the type of task, and (iii) the user profile.

Table 14. Framework for Evaluating Ubiquitous Applications [Sch04]

UEA	Metric	Conceptual measures
Attention	Focus	Number of times a user must change focus due to technology; number of displays/actions users need to accomplish, or to check progress, of an interaction; number of events not noticed by a user in acceptable times
	Overhead	Percent of time user spends switching among foci; workload imposed on user attributable to focus
Adoption	Rate	New users/unit of time; user rationale for using the application over an alternative; technology usage statistics
	Value	Changes in productivity; perceived cost/benefit; continuity for user; amount of user sacrifice
	Cost	User willingness to purchase technology; typical time spent setting up and maintaining the technology
	Availability	Number of actual users from each target user group; technology supply source; categories of users in post-deployment
	Flexibility	Number of tasks user can accomplish that were not originally envisioned; user ability to modify as improvements and features are added
Trust	Privacy	Type of information user has to divulge to obtain value from application; availability of the user's information to other users of the system or third parties
	Awareness	Ease of coordination with others in multi-user application; number of collisions with activities of others; user understanding about how recorded data is used; user understanding inferences that can be drawn about him or her by the application
	Control	Ability for users to manage how and by whom their data is used; types of recourse available to user in the event that his or her data is misused
Conceptual Models	Predictability of application behavior	Degree of match between user model and behavior of application
	Awareness of application capabilities	Degree of match between user's model and actual functionality of the application; degree of match between user's understanding of his or her responsibilities, system responsibilities, and the actual situation; degree to which user understands the application's boundary
	Vocabulary awareness	Degree of match between user's model and the syntax used by the application
Interaction	Effectiveness	Percentage of task completion
	Efficiency	Time to complete a task
	User satisfaction	User rating of performing the task
	Distraction	Time taken from the primary task; degradation of performance in primary task; level of user frustration
	Interaction transparency	Effectiveness comparisons on different sets of I/O devices
	Scalability	Effectiveness of interactions with large numbers of entities or users
	Collaborative interaction	Number of conflicts; percentage of conflicts resolved by the application; user feelings about conflicts and how they are resolved; user ability to recover from conflicts
Invisibility	Intelligibility	User's understanding of the system explanation
	Control	Effectiveness of interactions provided for user control of system initiative
	Accuracy	Match between the system's contextual model and the actual situation; appropriateness of action; match between the system action and the action the user would have requested
	Customization	Time to explicitly enter personalization information; time for the system to learn and adapt to the user's preferences
Impact and Side Effects	Utility	Changes in productivity or performance; changes in output quality
	Behavior changes	Type, frequency, and duration; willingness to modify behavior or tasks to use application; comfort ratings of wearable system components
	Social acceptance	Requirements placed on user outside of social norms; aesthetic ratings of system components
	Environment change	Type, frequency and duration; user's willingness to modify his environment to accommodate system
Appeal	Fun	Enjoyment level when using the application; level of anticipation prior to using the application; sense of loss when the application is unavailable
	Aesthetics	Ratings of application look and feel
	Status	Pride in using and owning the application; peer pressure felt to use or own the application
Application Robustness	Robustness	Percentage of transient faults that were invisible to user
	Performance speed	Measures of time from user interaction to feedback for user
	Volatility	Measures of interruptions based on dynamic set of users, hardware, or software

Lum and Lau (2002) believe that when user-specific and device-specific requirements are considered for content adaptation, the user satisfaction is improved (e.g. by creating different views of the same content for different users according to their preferences, in a user-centric approach).