A Computational Framework for Context-aware Adaptation of User Interfaces

Vivian Genaro Motti and Jean Vanderdonckt
Université catholique de Louvain, Louvain School of Management
Louvain Interaction Laboratory, Place des Doyens, 1 – B-1348 Louvain-la-Neuve (Belgium)
{vivian.genaromotti, jean.vanderdonckt}@uclouvain.be – Phone: +32 10 478525

Abstract— **In order to address challenges posed by different users conducting their interactive tasks on heterogeneous platforms and devices in various environments, this paper provides a computational framework to support the adaptation of the user interface of interactive systems. This framework consists of: a meta-model for understanding fundamental concepts required by adaptation, a reference framework for characterizing seven dimensions for conducting adaptation based on the meta-model, and a design space for consistently assessing the adaptation coverage. In this way, development phases are considered with a standard approach, a unified terminology, and an extensive catalog of techniques.**

Keywords — context-aware adaptation, user interfaces, computational framework

I. Introduction

To interact with computational systems, users with different profiles and in distinct environments employ devices ranging from feature phones to wall displays. Environments vary concerning their noise, light and stability levels, etc. Users' profiles also vary significantly. Thus, there is a considerable heterogeneity of contexts in which users interact [1], challenging stakeholders during the development of information systems. The user interfaces of such systems must be able to properly consider particularities and constraints of each context of use to not damage or even prevent, the user interaction. However, implementing one dedicated version of each system for each situation is neither feasible nor scalable. Based on context information [2,3], *context-aware adaptation* (CAA) adapts certain properties of an interactive system in order to improve the user interaction, initiated by the user (*adaptability*) and/or the system (*adaptivity* or *adaptiveness*).

In a scenario in which the conventional context of use, of an able-bodied user with a desktop PC in a stable environment is no longer valid, context-aware adaptation permits dealing with the heterogeneity of situations in which users interact. When the context information [2,3] is correctly considered while implementing systems, the usability and user satisfaction levels during interaction improve. Context-awareness is an essential design requirement to create systems that are more human-centered [4]. Although the adaptation of information systems aims at higher usability levels, there are different contexts to consider and different techniques to adapt systems' properties, thus, correct priorities for each context information and adaptation technique must be properly assigned.

Many works have been dedicated to advance the adaptation domain. Frameworks [1,4,5,6,7,8,9,10,11,12,13,14], surveys [15,16,17], methods and techniques [18,19,20,21], principles [22,23,24,25], and application domains [26] have already been proposed. However, these information sources are scattered, making it hard to find a unified source of guidance while developing interactive systems that support context-aware adaptation. Moreover, technology has quickly evolved, and the fragmented device market is challenging for developers.

To guide stakeholders with a unified terminology and possible approaches in this domain, we created a computational framework for context-aware adaptation (TriPlet). TriPlet includes a meta-model, a reference framework, and a design space. The meta-model (CAMM) consistently defines related concepts, properties and relationships, establishing a common ground for implementing adaptation. The reference framework (CARF) presents alternatives for adapting systems, i.e. how to do it, when, and why. The design space (CADS) is an analytical instrument to assess and to compare adaptation levels of different applications based on unified criteria.

This paper is organized as follows: Section 2 motivates this research, presents and discusses related works, Section 3 details the computational framework and its components, Section 4 applies the results and Section 5 concludes this work.

II. State of the Art

The variety of computational devices [1], summed with their consequent pervasiveness, mobility, and ubiquity [4] challenges the development of interactive systems [27]. Mainly because, it is hard to correctly identify relevant information coming from varied contexts in which users interact. Moreover, each context can dynamically vary, forcing user interfaces to be accordingly adapted accommodating specific characteristics and constraints of each context. Context-aware adaptation raises as a solution to prevent stakeholders from creating dedicated versions of the same system for each context's particularity. To better identify and consider context information and to properly execute adaptation, many studies have been performed, significantly advancing the development and the research in this field. Theoretical frameworks, surveys, applications domains, adaptation techniques, methods, strategies, approaches and principles have been defined and investigated since the early 90's. Such studies proved that multiple domains can actually benefit from adaptation. This section presents a selection of works that focus on meta-models, frameworks and design spaces for CAA, mainly because these concepts are the basis for TriPlet components, providing a common ground and inspiration for them.

A. Meta-models

Models abstract system concepts, their properties and relationships. In the CAA domain, models have been used to represent: context information, adaptation rules, and multimodal properties. [6,23,28,29] cover adaptation rules, [23] focusing on plasticity, and [30] targets at mobile devices.

In *Munich Reference Model* [31] defines 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.

In *ADAPTS* [19] explicitly models task, domain and users, in an integrated manner to support adaptation based on the context. A diagnostic engine employs user and expert models to update the navigation, selecting the most appropriate tasks for the user based on pre-defined weights.

For *Context Information* [32] provides a meta model to define context information and its associations. The main concepts considered include: devices, persons, and their properties, (mobile phone, phone number, gender, etc.) and their relationships, (is located nearby, has phone number, etc.).

For *Rules* [23] states that an adaptation model specifies the evolution and transition rules applied in context's changes. Their adaptation models define tasks, abstract, concrete, final UIs and widgets extensions, plasticity is seen as the main principle. They remark the benefits of using model-based approaches to implement CAA, and emphasize the adoption of principles, as: plasticity and continuity.

For *Mobile Applications* [30] defines a MOF-based model for context-aware mobile applications. The concepts considered are abstract including: classifier, attribute, entity, content, association, dependency, constraint and group.

The *Adaptation Rules* meta-model of [28] defines adaptation rules and targets at plasticity as the goal for ubiquitous applications. It aids designers to take decisions and implement CAA considering three phases: context perception, reaction, and 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.

The *Adaptation Rules* of [6] defines in a meta-model basic concepts, of adaptation rules as: precondition, event, sensor, data, transformation and rules.

The *Context of Use* is covered in [33] with generic model created for Morfeo project, defining element, property, entity, aspect, component, characteristic, environment and user.

Table 1 summarizes the works presented above and highlights their goals. They can be broadly organized in two groups: while [6,23 and 28] focus on rules, [32 and 33] focus on context. More specifically [19] focuses on user models and [30] targets 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 process, i.e. by focusing in one specific part of the process, a global definition is still missing.

Table 1. Meta-models for Context-aware Adaptation and their main goals

Meta-Model	Main Focus
Munich Reference Model [31]	User models (preferences, tasks, goals, experience) for adaptation, includes also rules
ADAPTS [19]	Task, domain and users. Tasks are then selected based on the user model.
Context Information [32]	Defines rules, context in terms of device and person, quality and associations,
Rules [23]	Evolution and transition rules based on context's changes. MBUI and plasticity are considered too.
Mobile [30]	Context-aware mobile applications.
Rules [28]	Adaptation rules for ubiquitous computing.
Rules [6]	Rules in terms of conditions and transformations
MORFEO [33]	Context-awareness and the users' profile.

B. Frameworks

Because there is no unique definition of framework, the ones that have been reported in the literature so far include, application toolkits [2,34], architectural approaches [1,9,35, 36], conceptual definitions [4], logics [8,37], etc. Thus often, they complement or specialize each other, making it hard to consistently analyze and compare them. This section summarizes existing frameworks that support CAA.

A Framework for Adaptable Hypermedia Documents (FAHD) [38] 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 standards 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. Presentation's specifications can be recorded in style sheets, and are broad enough to cover many hypermedia document sets.

Conceptual Framework for Adaptive Web Sites (CFAWS) [39] focus 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 improves navigation by making it more efficient.

Context-aware frameworks and toolkits (CaFT) [2] defined 'context information' as relevant concepts that characterize the user's context. They defined requirements for implementing context-aware systems, and created a framework that eases their implementations [3].

Personal Universal Controller (PUC) [36] creates an architectural framework composed of: appliance adaptors, communication protocol, specification language and interface generators. It controls real appliances and uses decision trees to render user interfaces in varied modalities. It enables users to control any appliance within their environment. The UI is automatically generated. The description of the appliance's functions is used as input. With user studies they noted that users preferred automatically generated UIs (instead of the manufacturer's UIs of the actual appliances).

W3C Multimodal Interaction Framework [35] presents a general architecture, involving major components and their respective functions for multimodal systems. Modalities include: speech, handwriting, keyboard and mouse. Considered languages: XHTML, SVG, SMIL and HTML. Input modes (recognition, interpretation and integration) and output modes (rendering, styling and generation) are considered. Use cases illustrate instantiations of this work.

A Framework for Adaptive Educational Hypermedia Systems (AEHS) [40] defines 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 presentation and navigation changes), and presentation generators (generating adaptation results).

FAÇADE [18] 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 user preferences. A distributed architecture separates context processing from content adaptation for ensuring flexibility and extensibility.

SUPPLE [34] is a framework-toolkit that treats UI generation as an optimization problem. It considers device's constraints and users' efforts. Its uses declarative descriptions of a UI, device characteristics, widgets, and a user and device cost function. For [34] an adaptive UI requires 3 inputs: the UI specification, a device model and a user model. SUPPLE includes as input a trace of typical user behaviour, enabling user-specific renderings.

ResOurce-aware Application Migration (ROAM) [12] is an application framework that assists developers in implementing applications to run in multiple devices, and that enables users to migrate applications across devices without much efforts. ROAM follows as adaptation strategies: transformations, dynamic instantiation and offloading computation. Agents support the migration. ROAM considers as context information just device properties, as: display size, input method and user interface library.

XUL-based Interface Framework (XIF) [37] separates the UI adaptation from the logic to ease the development of mobile apps, assuring more portability for Java ME settings.

PersonisAD [10] is an architectural framework to model and use context. Its key concern is scrutability, i.e. the users can access and understand their models using operations as access, tell, 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.

Architectural Framework for Automated Content Adaptation to Mobile Devices (ACAMD) [9] considers as basic requirements for adaptation frameworks: transforming images and identifying the delivery context. Aiming at cost-efficient development of mobile applications, they provide a markup language and an integrated development environment (IDE). ACAMD supports: navigation, organization, image conversion, data integration, fragmentation, layout and style.

Context-Aware Workflow Execution Conceptual Framework (CAWE) [7,8] enhances the flexibility of the workflow in web service composition systems by explicitly representing context information and adaptation rules in the adaptation logic. CAWE manages context-aware applications by hierarchically representing the workflow, it supports the execution of alternative actions' courses and the context-aware invocation of web services. It considers the UI adaptation and the workflow execution, and it can be extended to handle complex adaptation rules.

MIMOSA [1] 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) [5] 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. The layouts consider the platforms, e.g.: a default one (of 992 pixels, for desktops, laptops, and tablets in landscape orientation), a tablet one, mobile devices, and wide mobile one (for large mobile devices or landscape-oriented smartphones). The layouts vary their columns and margins.

Conceptual Framework (CF) [4] 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.

Table 2 presents the frameworks analyzed based on the context of use that they target (user, platform or environment), their main contributions (architecture, algorithms, models, toolkit, 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, it is classified as '-'. As also pointed by [1], 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 [3,7,8,1]. Still when the context is considered, the contextual information concerning the platform is prioritized instead of the user [36,35]. Moreover, as points [9] the adaptation frameworks' complexity leads to rejections of adaptation methods offered.

3) Design Spaces for Adaptation

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. [41] defines 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.

Table 2. Current frameworks according to their: contextual dimensions (user, platform and environment), support provided and 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 [38]	+	+	-	Framework, Architectural Forms	+	-	+
CFAWS [39]	+++	-	-	Algorithm, Methods	-	++	-
CaFT [3]	+++	+++	+++	Conceptual Framework, Toolkit	+	-	+
PUC [36]	-	++	-	Architecture, Specification, Language, UI Generator	++	-	-
W3C [35]	+	++	+	Generic Meta-Architecture	++	+	+
AEHS [40]	+++	-	-	Decision Models, Analysis, Strategies,	++	+++	+
FAÇADE [18]	+	+++	-	Architecture, Decision Engine, Context Repository	++	++	++
SUPPLE [34]	+	++	-	Framework-Toolkit	++	+	-
ROAM [12]	-	+++	-	System, Application Framework	++	+	-
XIF [37]	-	+++	+	Logical Approach, XUL-based UI Framework	+++	-	-
Personis AD [10]	+	+	+	Rule Language, Generalized Architectural Framework	++	-	++
ACAMD [9]	+	+++	+	IDE, Architecture	++	++	+++
CAWE [7,8]	+++	++	+	Framework, Architecture, Logic	++	++	+
MIMOSA [1]	++	+++	+	Architecture, Distributed Framework	++	++	+++
LF [5]	-	+++	-	Layout Options	+++	-	-
CF [4]	+++	+	++	Conceptual Framework	+++	+++	+++

The design space deals with tasks at the granularity of commands, aims at identifying software implications and constraints during its development phases, and it 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.

[42] defines 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).

[43] creates a design space for context-aware UIs. Containing adaptivity, adaptability, and context-awareness, axis including the target ('*with respect to what*'), aspects ('*what*'), qualities ('*for what*'), agents ('*who*'), amount ('*how many*'), temporality ('*when*'), and approach adopted ('*with what*'), such a design space encompasses seven relevant dimensions for context-awareness [43]. Such design space proposes a new method for developing context-aware UIs. It aids designers to locate, identify and separate events that change context and thus reconfigure the UIs.

[34] proposes 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 UIs' spatial stability. They also believe that frequent adaptations may reduce the utility of adaptive UIs.

[22] presents a dimension space that enables classification, comparison and contrasting different works on meta-UIs. Dimensions encompassed include: interaction techniques (integration, extensibility, representation, function) qualities (initiative and control) and functional coverages (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 [22] models and mechanisms are currently being developed for plastic and context-aware adaptive UI's. However care must be taken to ensure that end users have enough UI control.

For [44] 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).

[45] focuses 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 UIs' 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, defining the mapping between interaction options and generation of local digital footprints. Adaptation types were analyzed and linked with each digital footprint.

Table 3 presents the works on design space based on their dimensions and granularity levels. Dimensions include discrete and continuous values, and their levels not always represent a scalar relationship, although often they are represented

Table 3. Existing design spaces, their dimensions and levels

Design Spaces	Dimensions (levels' sample)
Nigay and Coutaz [41]	Modalities (sequential, parallel) Fusion (combined, independent) Abstraction (meaning, no meaning)
Karagiannidis [42]	What/constituents (semantics, syntatics, lexic) When/determinants (UI's states) Why/goals How/rules
SIMILAR DS [43]	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)
Gajos et al. [34]	Costs x Benefits (low, moderate, high) Frequency x Predictability (less, most) Performance x Satisfaction
Coutaz [22]	Technique (extensibility, design, representation, integration) Quality (initiative, control) Function (object types, generic services)
Arhippainen [44]	Target (user, environment, platform) Means (navigation, content, presentation) Time (static, dynamic/run-time)
Cardoso and José [45]	Presence (detection, characterization, identification, exposure) Content Suggestion Actionables

in continuous axes. Analogous to the frameworks reported, the design spaces are also dedicated to specific CAA aspects. While [45] tackles interactive displays, and [41] multimodal applications, [44] focuses on user experiences. All dimensions involved with these scenarios are relevant, but a global view of a CAA design space is still missing.

C. Shortcomings

Advancing specific adaptation topics in depth is important, but stakeholders lack a unified source in which they can rely to implement adaptation [46]. The main shortcoming in this domain is the lack of a unified approach for stakeholders, a standard terminology, and a common methodology. Once the information sources are scattered, developers either ignore adaptation [1], in a fast and often inaccessible solution, or spend significant efforts to look for and find information. Today various works must be analyzed to find right solutions for each case, requiring significant efforts and causing incompatible results. The definition of the TriPlet computational framework was inspired on works about CAA (models, languages, frameworks and softwares). Models are essential to define relevant concepts for CAA, and frameworks support CAA different phases, e.g. design, implementation and evaluation. The contributions of this work inherited a lot from works reported in this section, and TriPlet attempts to both: integrate and extend them. The solution described in this paper is innovative by presenting a computational framework that supports the software development life cycle in the incorporation of context-aware adaptation. TriPlet is also flexible and extensible. The concerns and shortcomings observed for CAA delineate the problem space for this work, and also lead to conclude requirements and improvements in this domain considering different dimensions.

III. TriPlet: A Computational Framework for Context-aware Adaptation

A framework is formally defined as a reusable, semi-complete structure that when specialized produces custom applications. It includes components extensible for specific domains. Frameworks are a proposition, which, if properly designed, reduce investments and development costs [9]. For [11], frameworks for context-aware computing ease the development and deployment of context-aware applications because stakeholders can focus on tasks that are more specific for their application, while relying on a basic structure to handle, manage and distribute information. A framework aids to create explicit structures, which can be made complete and comprehensive by repeated investigations over time. It also contributes to establish design guidelines and with a consistent terminology for sharing and describing results [47].

This section presents TriPlet and its development steps. First the literature was reviewed and adaptation concepts, as techniques, were systematically extracted. 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 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 design space with essential dimensions for analyzing and comparing multiple adaptability levels of CAA was defined.

A. Context-aware Meta-model

To formalize concepts for the CAA development, based on the results of the literature review, a context-aware meta-model was created (Figure 1). This model, named CAMM, uses the OMG notation for UML Class diagrams, being associations presented by named lines (e.g., triggers), aggregations presented by open diamonds (e.g., resource property), and compositions presented by closed diamonds (e.g., User). CAMM covers the complete adaptation process; it abstracts necessary concepts, establishes their relationships and defines their properties. Moreover, further information, as constraints and relations' cardinality are also specified.

Four colours are applied in the meta-model to separate concepts based on their specific domains. Thus, the classes represented in red refer to the adaptation agents, the green ones refer to the context of use, the yellow ones refer to the core of the adaptation process, and purple ones to the model generation.

This Meta-Object Facility (MOF) based meta model diagram illustrates with the red blocks possible agents to trigger an adaptation process: the system, the user or a third party, abstracted as 'Adapter'. Considering the several phases of an adaptation process, each agent can be responsible for each phase [24]. E.g., the end user starts the adaptation, and the system decides the best method among the ones available. Besides, the agent roles can be further refined based on their specific characteristics and interrelationships, supporting collaboration and hierarchies.

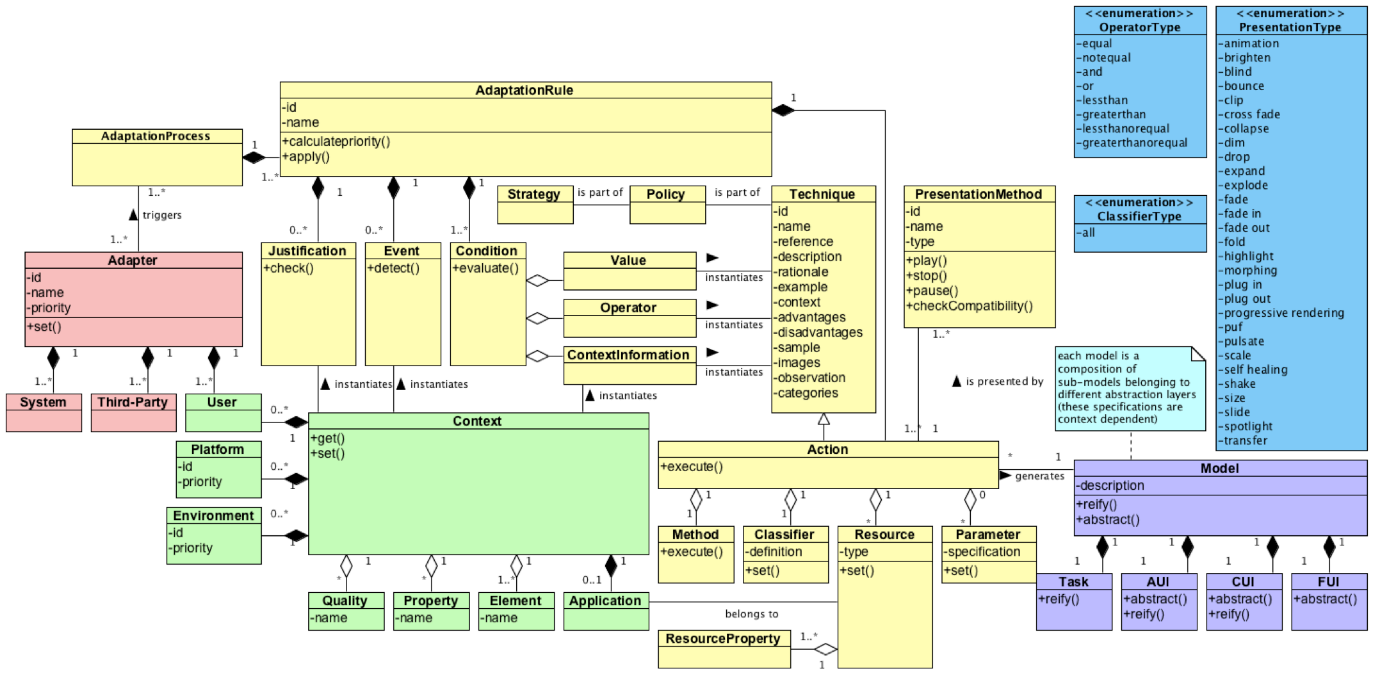


Figure 1. CAMM: Context-aware Meta Model

A CAA process can also be triggered by a change in the context of use. The green blocks in the meta-model diagram represent concepts related to the context information. The context defines adaptation rules by providing information to instantiate them. When the user changes the orientation of the device, a technique like 'change the UI orientation' must be applied, rotating the UI contents based on the new position of the device (information possibly gathered by a sensor).

As the context consists 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 rules. 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) [48], instantiated and triggered by context information. More than one rule is normally applied simultaneously, so conflicts may appear and adaptation must be progressively processed [46]. To solve them, priorities must be assigned for certain contexts: adaptation techniques may be composed as policies (meta-rules) that can also be composed as strategies (meta meta-rules). An extension of ECA rules that includes also Justification can be applied too.

CAA results can be presented to the end user with different methods, preventing the end user disruption, commonly caused by significant differences between the original and the adapted UI. Animation is one method that can be applied in this sense. By using animation, the intermediary steps of a transition are explicitly presented to the end user, for a more intuitive comprehension of sequential changes [49].

Rules actions generate models for SFE. In CAMM, models are presented by purple blocks, and based on principles of the model driven approach, they range from task and concept level, abstract level, to concrete and final level [13]. While a task model specifies tasks and subtasks involved in accomplishing a user goal, the final UI level defines the layout (e.g. for GUIs): style, alignment, and colours.

B. *Context-aware Reference Framework*

The Context-aware Reference Framework (CARF) is a reference framework that lists the most relevant concepts for implementing and executing CAA. The CARF, whose center is illustrated in Figure 2, is graphically represented by a mind map and composed by seven central branches. While these central branches (i.e., the ones directly connected to the root) present abstract concepts, the more external ones (added under the central ones) list possible instances for these abstract concepts, aiding the implementation, execution and analysis of CAA. 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>*. In natural language, e.g. it could mean: at run time concerning the user age the system client must simplify the textual content to improve (or assure) its accessibility. The seven central branches of the CARF refer to, in clockwise sense: what, why, how, to what, who, when, and where dimensions, being defined as follows:

- **What:** the type of resource or aspect that is adapted, including three main categories [19,42,43,44]: navigational flow, presentation or content. E.g.: images or text;
- **Why:** the main adaptation goals, expressed as software qualities [34,23,50,51]. E.g.: adaptation performed targeting at better usability levels;
- **How:** in which way the adaptation is performed, methods, techniques and strategies for the adaptation [42,51]. E.g., technique of changing the video quality (see Figure 3);
- **To what:** contextual information to justify and define the adaptation, i.e., application resources subject to adaptation

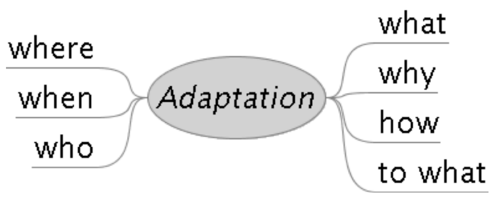


Figure 2. CARF: Context-aware Reference Framework

based mainly in user, platform, or environment. E.g.: adapting to color-blind users [2,3,27,51,52];

- **Who:** refers to the actor who triggers, initiates or is in charge of each phase of the adaptation, e.g.: the end user, the system, or a third party. In a mixed approach both users and system collaborate in the adaptation [6,43,51];
- **When:** the state in which the adaptation process is performed, i.e., design time, run time, compilation time. E.g.: adaptation performed at run time [6,28,42,43,44];
- **Where:** the 'location' in which the adaptation takes place, i.e., based on the architecture adopted it can be at the client, at the proxy, or at the server [6,51]. E.g.: adaptation performed at the server side.

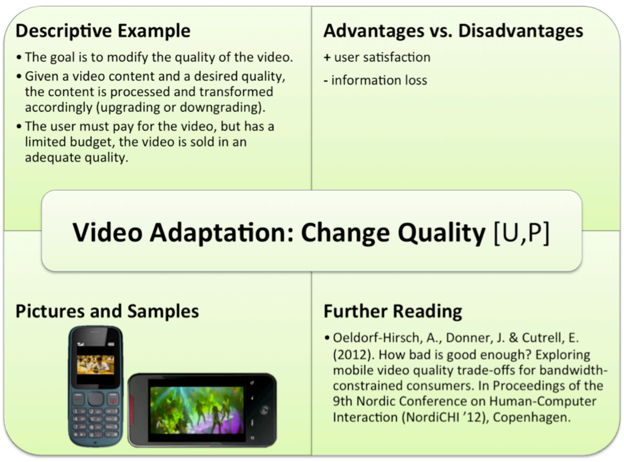
While adapting the resources and their properties, an interactive system is modified, by including, editing, removing, or simplifying them being such changes named as *adaptation techniques*. With a literature review, more than 150 different techniques were catalogued[1]. They are organized as cards with detailed information: references, definitions, benefits, context, etc. These cards (see Figure 3 example) guide stakeholders while retrieving specific information about adaptation, e.g. concerning all adaptation techniques for small screen devices and color-blind users. Adaptation techniques are the 'brick' of an adaptation process being thus the atomic unit that when combined result in a whole adaptation process. To compose a complete adaptation process further concepts are also needed.

The seven branches of CARF compose its core, and by adding new instances they can be refined, however they should not be extended with additional branches (once these concepts were already selected as the most essential for characterizing this domain, being them sufficient to comprise and express all necessary phases of a CAA process). The CARF defines the most relevant concepts for CAA and extensively lists and presents possibilities for implementation and execution. CARF can be used before the implementation phase of an application, as an extensive catalogue to guide developers to take design decisions, or after the implementation of an application, to analyze and evaluate concepts that were considered, identifying underexplored areas in which future extensions are possible.

C. *Context-Aware Design Space*

Design Spaces (DS) guide stakeholders to take better decisions during a project, aiding to select relevant aspects based on projects' goals. Besides this, with a later analysis stakeholders can assess and extend a project development. During the development life cycle, stakeholders can also use a DS to update project requirements and to identify possible alternatives. A DS helps to communicate design decisions by enabling their documentation, their future reviews (verify the features available), and also to compare applications.

[1] At: http://tiny.cc/xye0uw


Descriptive Example
- The goal is to modify the quality of the video.
- Given a video content and a desired quality, the content is processed and transformed accordingly (upgrading or downgrading).
- The user must pay for the video, but has a limited budget, the video is sold in an adequate quality.

Advantages vs. Disadvantages
+ user satisfaction
- information loss

Video Adaptation: Change Quality [U,P]

Pictures and Samples

Further Reading
- Oeldorf-Hirsch, A., Donner, J. & Cutrell, E. (2012). How bad is good enough? Exploring mobile video quality trade-offs for bandwidth-constrained consumers. In Proceedings of the 9th Nordic Conference on Human-Computer Interaction (NordiCHI '12), Copenhagen.

Figure 3. Cards for describing Adaptation Techniques

The main challenge for defining Design Spaces is selecting precisely descriptive dimensions for most of the applications, and significant granularity levels, to accommodate all possible decision, assuring still enough legibility to locate and identify them, and being as much extensive and precise as possible.

The Context-aware Design Space (CADS) (Figure 4) is a theoretical method to supports stakeholders in implementation phases, and in the analysis and evaluation of adaptive and adaptable applications. The CADS aids developers before and after the implementation phases. Before it, CADS aids to identify possible dimensions and granularity levels for performing adaptation, and after it CADS aids to analyze, evaluate and compare these dimensions regarding their respective coverage levels. Thus the CADS supports the analysis and the comparison of different applications that execute adaptation and during their complete development life cycle. It has been built in an iterative manner. First, relevant dimensions of CAA were identified based on the literature review. Then the specific granularity levels for each dimension were defined. In a first version of the diagram, because not all dimensions represent ordered values, the diagram was misinterpreted. As a result, CADS was split. Dimensions that are unordered belong to the CARF (e.g. context information). Solely dimensions that are ordered were kept, e.g. the applicability level for CAA (ranging from the entire application, to specific properties of the UI elements).

As a radar chart, the CADS is 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, qualitative values are represented with their respective empirical scale associated. The CADS considers the benefits towards the virtues proposed by [53] for design spaces, being thus *comparative* since multiple applications can be analyzed based on the same criteria; *exploratory*, since each dimension can be analyzed in terms of exploration, i.e., identifying further opportunities for extensions; and *descriptive*, since each dimension is precisely defined, consistently and uniquely. CADS is extensible, once its dimensions can be added or refined and flexible, once they can also be removed or added enabling focused analyses.

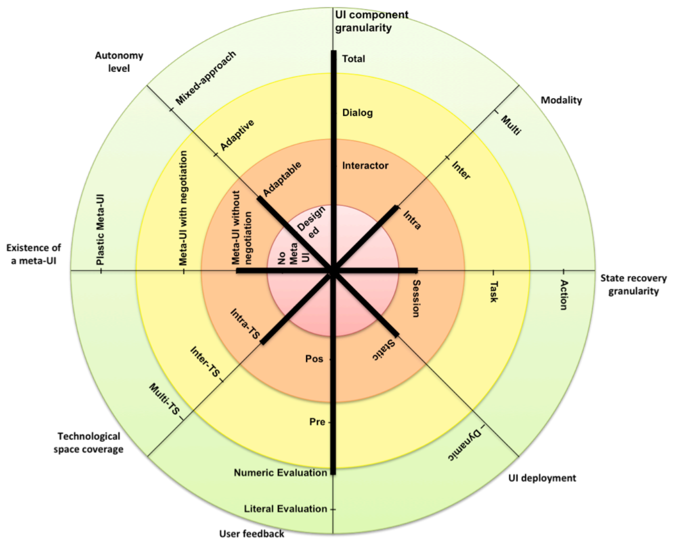


Figure 4. CADS: Context-aware Design Space

Clearly, the interpretation of scales for the dimensions chosen can vary based on the context. However, it is a general interpretation that is assumed for CADS. Once the concepts cannot (in principle) be numerically assessed and compared, their semantic meanings and interpretations must be considered. The proportions are also empirically associated with the dimensions, since no formal experiments were conducted so far to identify actual metrics for each dimension and its granularity levels. For each case of CADS application, its use must be defined and discussed. All CADS dimensions, although comprised in the same representation, are still independent, and thus concentric circles while aid the visual comparison of different granularity levels, do not necessarily represent same coverage levels between different dimensions.

The central circle of the CADS, colored in red, represents the absence of adaptation features, e.g. when no adaptation process is performed an application can be classified as designed based on its autonomy level. For each subsequent circle an additional coverage level of adaptation can be considered added, and more external levels represent higher coverages based on one adaptation dimension. So, supposing an application able to adapt for varied modalities (multi), it can be classified as having a higher coverage level of adaptation regarding the modality dimension if compared with another application that performs adaptation just within the same modality type (intra). A higher coverage level of adaptation regarding one specific dimension does not immediately imply a higher level of usability or a better application for the end users though. 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 continuous iteration of evaluation and improvements, and thus while it maintained the benefits of its previous version, it discarded 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, and thus dimensions can be removed, inserted, or refined. Below there is a list of the dimensions included in the complete CADS. Clearly, for more focused analyses, a specific set of 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 dimensions described below present the basic structure for the CADS. The scales' sizes for each dimension level are arbitrarily defined:

• **User Interface Component Granularity:** defines the abstraction levels for UI elements that can be subject to adaptation. Three levels are defined for these dimensions, interactor, dialog and total. Interactors correspond to UI elements (e.g., a combobox), dialog refers to containers (i.e., a UI elements composition), and total level refers to CAA that impact the complete window. Such examples are mainly applied in the context of GUIs 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 for the user interaction, when the same modality is maintained 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 obliged to quit the session and restart 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. 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 the email box, logging in, and so on (thus including both task and action).

• **User Interface Deployment:** represents how much adaptation has been pre-defined at design-time vs. computed at runtime, 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 user opinion is considered, 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, providing further details about their feedback.

• **Technological Space Coverage:** 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.

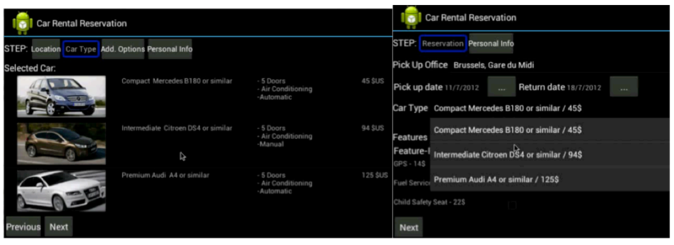

Car Rental Reservation
STEP: Location Car Type Add. Options Personal Info
Selected Car:
Compact Mercedes B180 or similar
- 5 Doors
- Air Conditioning
-Automatic
45 $US
Intermediate Citroen DS4 or similar
- 5 Doors
- Air Conditioning
-Manual
94 $US
Premium Audi A4 or similar
- 5 Doors
- Air Conditioning
-Automatic
125 $US
Previous
Next
Car Rental Reservation
STEP: Reservation Personal Info
Pick Up Office Brussels, Gare du Midi
Pick up date 11/7/2012
Return date 18/7/2012
Car Type Compact Mercedes B180 or similar / 45$
Compact Mercedes B180 or similar / 45$
Intermediate Citroen DS4 or similar / 94$
Premium Audi A4 or similar / 125$
Features
GPS - 14$
Child Safety Seat - 22$
Next

Figure 5: First implementation, context (i) and (ii) illustrating the car selection

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 to formally represent and handle adaptation. Users may control, assess and evolve it, including: no-meta UI, meta-UI without negotiation, meta-UI with negotiation, and plastic meta-UI.

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

To apply the CADS, the main axes are used to mark the coverage level for each dimension. The extension of the marks is defined based on what the application offers as adaptation. This permits a graphical visualization of the coverage level that is available. So, 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 each dimension. 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 considered to later on to extend or improve the application adaptation. Other option to mark dimensions consists in coloring (with stronger tones) the region of the circle under the interest level, however this approach works well only if all the levels correspond to the circles, and besides comparing multiple applications would not be possible with this approach.

To compare two or more applications, developers have two choices: (i) parallel lines can be drawn in different colors, enabling 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 simultaneously compared, however for a large number of samples the second approach is preferred, not affecting the readability of the dimensions' labels. Once the comparison of multiple applications rely on color to differentiate them, it is necessary to choose then different tones or styles, thus avoiding accessibility issues that may rise for example for color blind users. Figure 4 illustrates an applied CADS to analyze a given 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 developers to analyze dimensions based on their needs, i.e., they can select which dimensions will be considered in the analysis and use the diagram applying only dimensions of interest. A CADS version applied considering 6 dimensions permits developers to perform more fine-grained analysis. Besides this, the CADS is flexible and extensible, accommodating further dimensions and levels. The main criteria to perform this consists in assuring that it is still possible to analyze the dimensions in a ordered way, e.g. 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 able to take decisions during the adaptation process.

CADS current version results of improving previous versions; its weaknesses and strengths were analyzed and discussed during project meetings, presentations and also with internal surveys. Details about the evolution process of CADS are described in D2.1.25 [54]. The current version maintains strong points of preliminary versions and overcomes misunderstandings caused by unordered dimensions. To solve this issue, these dimensions were transferred to the CARF. Although the CADS establishes an empirical relation of order among levels that compose each dimension, the concepts considered are still linked with qualitative variables, so in principle they cannot be numerically evaluated and proportionally compared. Thus, for each application it is necessary to justify the selection process and its respective usage. CADS main benefit is analyzing adaptation in a unified and graphical view, simultaneously considering relevant dimensions and levels for a context.

IV. Framework Applications: Case Study

This section presents how to apply TriPlet in development phases of interactive systems. First a common case study is described, then, three possible instantiations are detailed.

The case study is a car rental example, in which users set specifications about the rental (e.g. period, place) and the car (type, fuel, extra's). This case study is merely illustrative, given that all domains can benefit of adaptation. However it was chosen as a basic example to illustrate the usefulness and benefits of TriPlet by encompassing varied context dimensions and application aspects targeted by the adaptation. To show multidimensional aspects of the framework, contexts varying in user, platform and environment were considered, also adaptations that impact presentation, navigation, and contents.

The case studies permit to cross-validate the theoretical concepts of the framework, assuring that it is enough comprehensive to cover all phases of the development life-cycle and also different requirements.

A. First Implementation

A tablet PC running Android was chosen as the platform, and 2 use cases were defined (Figure 5): (i) users without experience with car rental applications, medium experience with mobile devices, in a calm and stable environment (i.e. no loud noises, no stressful situation), and (ii) users experienced with car rental applications, and with mobile devices, and located in

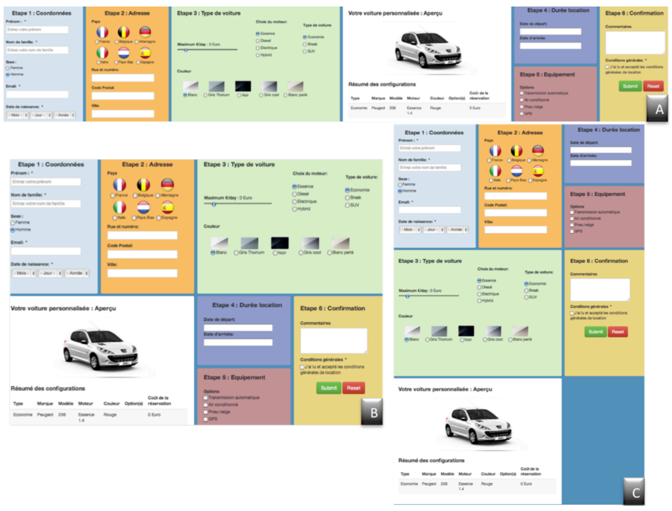


Figure 6: Second implementation, context A (horizontally aligned), B (balanced layout) and C (vertically aligned)

a stressful environment with a short time to conclude the task. Given that the platform is the same for both cases (tablet), a meta-rule was implemented. Tablet devices have a limited screen dimension and input controls (no mouse, or keyboard available). First, the Android guidelines must be respected (providing immediate feedback for user' touches by highlighting selections). Then more specific rules were defined: beginners must clearly see the interaction steps (explicitly indicated), the amount of information displayed is limited (avoiding cognitive overload) and the UI elements must be intuitive and simpler. For calm environments, each interaction step can provide its detailed information, the main task can be split in many sub-tasks, and the UIs target at specific actions.

B. *Second Implementation*

For the second implementation the context for the car rental example consider screen dimensions and resolution. The layout is automatically and progressively adapted to fit the contents in all space available, minimizing scrolling (Figure 6). jQuery Masonry plugin arranges UI components according to the re-size of the browser. Each component is treated individually, and moves to another column (or row) of the layout to fit according to new browser size. Thresholds assure the layout balance, 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 reorganization is automatic and progressive. Any screen dimension can be considered, due to fine-grained adjustments of the layout based on the browser size. Three adaptation techniques compose the CAA rules used: (i) resizing elements: scaling

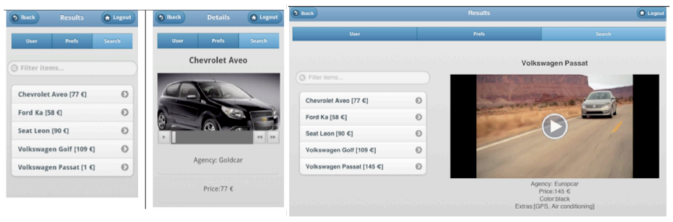


Figure 7: Third implementation: for a smartphone and for a tablet PC

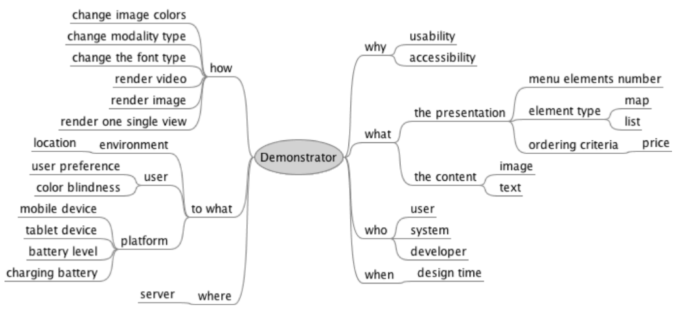


Figure 8: Instantiated CARF for demonstration

font size, UI elements as videos and images; (ii) reorganizing elements: changing the components horizontally and vertically to assure a balanced layout; and (iii) mixed approach: resizing and reorganizing. The instantiation of CAA rules conditions vary proportionally based on the browser window size, i.e. the bigger the window, the bigger the UI elements and amount of columns and rows of the layout.

C. *Third Implementation*

The car rental example was also applied in a third scenario of CAA based on: the user visual impairment (color blindness), the platform type (mobile phone, tablet device) (Figure 7), its battery level, and user preferences (set in the system). Six adaptation techniques were chosen and implemented (e.g.: changing the modality and the image colors), aiming at good usability and accessibility levels, by adapting presentation (e.g.: menu elements), and 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. The CARF (complete version) was applied to specify this implementation example as a means of analyzing possible options for CAA. The CARF instantiation illustrates this example (Figure 8).

D. *Application*

The implementations of the case study consider different context dimensions: visually impaired users, tablet pcs, large screens, mobile phones, low battery level, etc. Different dimension levels were combined: devices (a mobile phone, a Tablet PC), environments (relaxed vs. stressful), and users' profile (experienced, color-blind users). For the implementations, the system specifications concerning CAA were based on the theoretical models. After the implementation, the CADS was applied to analyze the coverage levels of CAA of the scenarios and the exploration of the CADS dimensions.

Figure 9 illustrates the applied CADS. The blue, black, and gray axes represent respectively the analyses of the first, the second and the third implementations of the car rental examples. In this CADS it is possible to 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. Thus, by the instantiation of the framework components, one

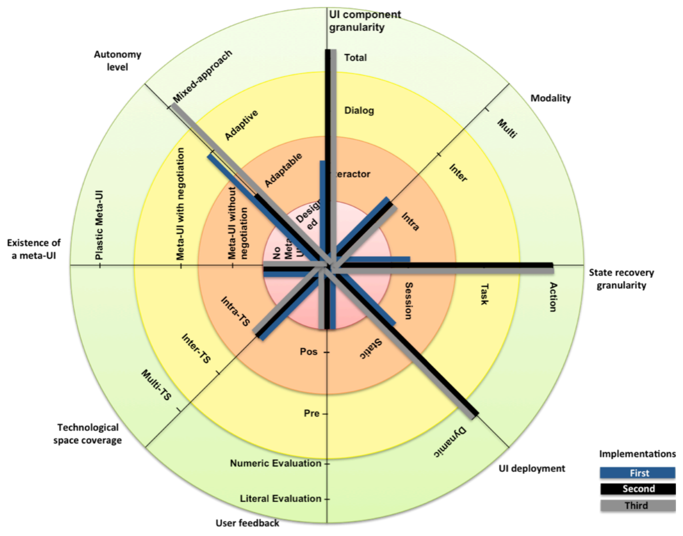


Figure 9: Instantiated CADS for demonstration

can analyze their applicability. Both CARF and CADS are useful for stakeholders to define the CAA process and to analyze the system in terms of adaptation levels. And the CAMM guides developers during the SDLC of a system, in special during its early stages (definition).

V. Final Remarks

Given the relevancy of providing CAA nowadays, due to device fragmentation, heterogeneous users, and exponentially growing applications, it is beneficial to have a framework on which stakeholders can rely to develop their applications. In this sense, TriPlet defines foundations for developing applications that perform CAA, i.e. by means of a computational framework, stakeholders of such applications can find support for all the development phases of information systems.

The contributions of this work are the result of an extensive and systematic review and analysis of the scientific literature regarding CAA. Such review resulted in an innovative general-purpose computational framework to support stakeholders during the complete SDLC of CAA, composed by three specific components: (i) a meta-model, CAMM that formalizes and abstract the main concepts (and their relationships) for implementing CAA; (ii) a reference framework, CARF that provides stakeholder support to define, specify and to decide the design for implementing CAA (CARF includes more than 150 templates composed by 11 fields describing adaptation techniques); and (iii) a design space, CADS that supports stakeholders in analyzing, comparing and evaluating the coverage levels of adaptation for context-aware applications. To evaluate a framework, either it is applied in different situations, i.e. considering different types of projects, industrial and scientific domains, different application domains and different complexity levels [11], or it is used to fit several published works by practicing researchers to frameworks in the studies, as propose [47]. In the case study, different contexts of use for a common application were defined using TriPlet.

Most of the related works consider a web-based context, although this fact can be considered as a limitation of the work, we believe that the contributions of this framework are generic and flexible enough in order to accommodate also applications that are not specifically web-based. Even considering an extensive list of related works and possible concepts, not all possible approaches can be covered at once; as such TriPlet is generic, extensible and flexible, aiming at a continuous update e.g. considering adaptation techniques and its application in heterogeneous domains and scenarios.

Working with CAA in a broad perspective is a challenge. However, it is also the most considerable gap in this domain, as such we highlight the importance of working with an overview of CAA domain to tackle its main issues. The case studies focus in heterogeneous contexts and aspects in an attempt to effectively validate the outcomes. Analogous to the work of [42], because the framework proposed is not hard-coded into a system and also not technology-driven is it flexible enough to enable attributes of an adaptation process to be modified and as such satisfy requirements for multiple application domains. Moreover, analogously to [45], because TriPlet establishes mappings between several context information and adaptation techniques, it covers adaptation needs without being specific to a particular scenario or domain, resulting in a generic context-aware method. As future works, experiments are planned to identify and analyze application costs, compared with current practices of development for context-aware adaptation.

Acknowledgment

This work was realized thanks to the support of Serenoa project, funded by the European Union through its 7th Framework Programme as a STREP Project: FP7-ICT-258030

References

[1] D. Malandrino, et al.. “MIMOSA: context-aware adaptation for ubiquitous web access,” in Pers Ubiquit Comput (2010) 14:301-320. Springer-Verlag London.

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

[3] A. K. Dey, G. D. Abowd, and D. Salber, “A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications,” in Human-Computer Interaction 16, 2. 2001, pp. 97-166.

[4] G. Fischer, “Context-aware systems: the 'right' information, at the 'right' time, in the 'right' place, in the 'right' way, to the 'right' person,” in Proc. of Int. Conf. on Adv. Vis Int. (AVI '12), ACM, USA, 2012, pp. 287-294.

[5] J. Korpi. Less Framework. At: http://lessframework.com/ (2012)

[6] V. López-Jaquero, J. Vanderdonckt, F. Montero, P. González, “Towards an extended model of user interface adaptation: the ISATINE framework,” In: Engineering Interactive Systems. Springer Berlin Heidelberg, 2008. p. 374-392.

[7] L. Ardissono, R. Furnari, A. Goy, G. Petrone, and M. Segnan, “A framework for the management of context-aware workflow systems,” in: Proceedings of the 3rd International Conference on Web Information Systems and Technologies (WEBIST'07). 2007. p. 80-87.

[8] L. Ardissono, A. Goy and G. Petrone. ‘‘A framework for the development of distributed, context-aware adaptive hypermedia applications’’, in: Adaptive Hypermedia and Adaptive Web-Based Systems. Springer Berlin Heidelberg, 2008. p. 259-262.

[9] B. Jankowska. “Architectural frameworks for automated content adaptation to mobile devices based on open-source technologies,” PhD Thesis. Europa Universität Viadrina Frankfurt, Germany, 2007.

[10] M. Assad, D. J. Carmichael, J. Kay and B. Kummerfeld, "PersonisAD: Distributed, active, scrutable model framework for context-aware services." Pervasive Comp. Springer Berlin Heidelberg, 2007. 55-72.

[11] J. E. Bardram, ""The Java Context Awareness Framework (JCAF)–a service infrastructure and programming framework for context-aware applications." Perv. Comp. Springer Berlin Heidelberg, 2005. 98-115.

[12] H. H. Chu, H. Song, C. Wong, S. Kurakake, and M. Katagiri, "Roam, a seamless application framework," Journal of Systems and Software, v. 69, n. 3, p. 209-226, 2004.

[13] G. Calvary, J. Coutaz, D. Thevenin, Q. Limbourg, L. Bouillon and J. Vanderdonckt, "A unifying reference framework for multi-target user interfaces," Interacting with Computers, v. 15, n. 3, p. 289-308, 2003.

[14] G. Calvary, et al., 2002 The CAMELEON Reference Framework, Deliverable 1.1, CAMELEON Project.

[15] J. Gómez and T. Tran. 2009, "A Survey on Approaches to Adaptation on the Web," Emerging Topics and Technologies in Information Systems, p. 136-152, 2009.

[16] R. Oppermann, "Adaptively supported Adaptability," International Journal of Human-Computer Studies, pp. 544 – 472, 1994.

[17] A. F. Norcio and J. Stanley, "Adaptive human computer interfaces: A literature survey and perspective," Transactions on System, Man and Cybernectics IEEE Transactions on, v. 19, n. 2, p. 399-408, 1989.

[18] B. Kurz, I. Popescu, and S. Gallacher, "FAÇADE – A Framework for context-aware content adaptation and delivery," In: Communication Networks and Services Research, 2004. Proceedings. Second Annual Conference on. IEEE, p. 46-55, 2004.

[19] P. Brusilovsky and D. W. Cooper, "Domain, task, and user models for an adaptive hypermedia performance support system," In: Proc. of the 7th int. conference on Intelligent user interfaces. ACM, p. 23-30, 2002.

[20] A. Lorenz, R. Oppermann, and A. Zimmermann. "Adaptive and Context-Aware Systems: A Survey." p. 22, 2000.

[21] P. Brusilovsky, "Methods and techniques of adaptive hypermedia. User modelling and User-Adapted Interaction," Special issue on: Adaptive Hypertext and Hypermedia, v. 6, n. 2-3, July 1996.

[22] J. Coutaz, "Meta-User Interfaces for Ambient Spaces," In: Proc. Of TAMODIA 2006 K. Coninx, K. Luyten, and K.A. Schneider, LNCS 4385, p. 1–15, 2007. Springer-Verlag Berlin Heidelberg 2007 pp. 1–15.

[23] G. Calvary, O. Daassi, J. Coutaz, and A. Demeure, "Des widgets aux comets pour la Plasticité des Systèmes Interactifs," Revue d'Interaction Homme-Machine, v. 6, n. 1, p. 33-53, 2005.

[24] E. Horvitz, "Principles of Mixed-Initiative User Interfaces," In: Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit. ACM, 1999. p. 159-166.

[25] H. Dieterich, U. Malinowski, T. Kühme, and M. Schneider-Hufschmidt, "State of the Art in Adaptive User Interfaces," in Adaptive User Int.: Principles and Practice, Schneider-Hufschmidt et al., p.13-48, 1994.

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

[27] G. D. Abowd, "Software engineering issues for ubiquitous computing," In: Software Engineering, 1999. Proceedings of the 1999 International Conference on. IEEE, p. 75-84, 1999.

[28] V. Ganneau, G. Calvary, and R. Demumieux, "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. ACM, p. 91-98, 2007.

[29] C.H. Muntean, G.M. Muntean, J. McManis, and A.I. Cristea, "Authoring Model for Quality of Experience-aware Adaptive Hypermedia Systems", In: Int. Workshop on Authoring of Adaptive and Adaptable Hypermedia (A3H 2006),2006, Dublin, Ireland.

[30] C. R. G. de Farias, M. M. Leite, C. Z. Calvi, R. M. Pessoa, and J. G. Pereira Filho, "A MOF metamodel for the development of context-aware mobile applications," In: Proceedings of the 2007 ACM symposium on Applied computing. ACM, 2007. p. 947-952.

[31] N. P. D. Koch, "Software engineering for adaptive hypermedia systems and development process," 2000. PhD Thesis

[32] F. Fuchs, I. Hochstatter, M. Krause, and M. Berger, "A metamodel approach to context information," In: Pervasive Computing and Communications Workshops, 2005. PerCom 2005 Workshops. Third IEEE International Conference on. IEEE, 2005. p. 8-14.

[33] Morfeo Project, (2012) Context of Use Meta model. Available online at: http://forge.morfeo-project.org/wiki_en/index.php/Context_Of_Use_ Metamodel#Introduction.

[34] K. Z. Gajos, M. Czerwinski, D. S. Tan, and D. S. Weld, "Exploring the design space for adaptive graphical user interfaces," In: Proc. of the working conf. on Advanced visual interfaces. ACM, p. 201-208, 2006.

[35] W3C Multimodal Interaction Framework. Larson, J., Raman, T., and Raggett, D. (2003) W3C Note. http://www.w3.org/TR/mmi-framework/

[36] J. Nichols, B. Myers, T. K. Harris, R. Rosenfield, M. Pignol et al.. "Generating remote control interfaces for complex appliances," in CHI Letters: UIST '02 Proceedings of the 15th annual ACM symposium on User interface software and technology, Paris, France, 2002.

[37] T. Butter, M. Aleksy, P. Bostan, M. Schader, "Context-aware user interface framework for mobile applications," in: Distributed Computing Systems Workshops, 2007. ICDCSW'07. 27th International Conference on. IEEE, 2007. p. 39, 22-29 June 2007 doi: 10.1109/ICDCSW.2007.31.

[38] R. Lloyd, et al. "A framework for generating adaptable hypermedia documents." Proceedings of the fifth ACM international conference on Multimedia. ACM, 1997.

[39] M. Perkowitz and O. Etzioni, "Towards adaptive Web sites: Conceptual framework and case study," in: Journal of Artificial Intelligence, v. 118, n. 1, p. 245-275, 2000.

[40] J.M.P. Oliveira, C.T. Fernandes, "A framework for adaptive educational hypermedia system," In: Workshop on Apps, Products and Services of Web-based Support Systems, IEEE/WIC, Halifax. Proc, 2003.

[41] L. Nigay and J. Coutaz, "A design space for multimodal systems: concurrent processing and data fusion," in: Proc. of the INTERACT'93 CHI'93 Human factors in computing systems. ACM p. 172-178 1993.

[42] C. Karagiannidis, A. Koumpis, and C. Stephanidis, "Deciding 'what', 'when', 'why', and 'how' to adapt in intelligent multimedia presentation systems," In: 12th European Conference on Artificial Intelligence. Budapest, Hungary, 1996.

[43] J. Vanderdonckt, D. Grolaux, P. Van Roy, Q. Limbourg, B. Marcq and B. Michel, "A design space for context-sensitive user interface," in Proceedings of IASSE, Keywords Challenges of Context-Sensitive User Interfaces DS for Context-Sensitive UIs," p. 207–214, 2005.

[44] L. Arhippainen, "Studying user experience: issues and problems of mobile services – Case ADAMOS: User experience (im)possible to catch?" Fac. of Science, Dep. of Inf. Proc, Univ. of Oulu, Finland 2009.

[45] J. C. S. Cardoso and R. José, "A framework for context-aware adaptation," Human Factors, pp. 118–127, 2009.

[46] A. Cristea, and L. Calvi, "The 3 layers of adaptation granularity," in Proceedings of the 9th International Conference on User modeling (UM'03), P. Brusilovsky, A. Corbett, and F. de Rosis. Springer-Verlag, Berlin, Heidelberg, p. 4-14, 2003.

[47] J. Scholtz and S. Consolvo, "Toward a framework for evaluating ubiquitous computing applications," Pervasive Computing, IEEE, v. 3, n. 2, p. 82-88, 2004.

[48] K. Dittrich, S. Gatziu, and A. Geppert, "The active database management system manifesto: a rulebase of ADBMS features," SIGMOD Rec. 25, 3 (September 1996), p. 40-49.

[49] Ch.-E. Dessart, V. G. Motti and J. Vanderdonckt, J, "Showing User Interface Adaptivity by Animated Transitions" In Proc. of 3rd ACM Symp. on Eng. Interactive Comp. Sys. EICS'2011. ACM, NY, 95-104.

[50] C. Stephanidis, A. Savidis, and D. Akoumianakis, "Towards user interfaces for all", in proc. of 2nd TIDE congress, pp. 167– 170, 1995.

[51] J. Rouillard, "Adaptation en contexte: contribution aux interfaces multimodales et multicanal," PhD Thesis, Univ. des Sciences et Techn. Lille, France, 2008.

[52] A. Zimmermann. 2007. "Context Management and Personalization: A Tool Suite for Context- and User-Aware Computing" Ph.D. thesis, Fakultät für Mathematik, Informatik und Naturwissenschaften der R-W.

[53] M. Beaudouin-Lafon, "Instrumental Interaction: An Interaction Model for Designing PostWIMP User Interfaces," In: Proc. of the SIGCHI conf. on Human factors in computing systems. ACM p. 446-453, 2000.

[54] V. G. Motti, (2012). Del. 2.1.2: CARF and CADS (R2). http://files.morfeo-project.org/serenoa/public/deliverables/M18/ SERENOA_D2.1.2.pdf