Multi-Dimensional Context-Aware Adaptation of Service Front-Ends

Project no. FP7 – ICT – 258030

Deliverable 3.1.1 Reference Models Specification (R1)

Due date of deliverable: 31/08/2011

Actual submission to EC date: 30/08/2011

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)		
Dissemination level		
[PU]	**[Public]**	**Yes**

Document Information	
Lead Contractor	UCL
Editor	Vivian Genaro Motti
Revision	26.08.2011
Reviewer 1	Telefónica I+D
Reviewer 2	
Approved by	
Project Officer	Jorge Gasós

Contributors	
Partner	**Contributors**
UCL	Vivian Genaro Motti
CNR	Davide Spano
TID	Javier Caminero Gil

Changes			
Version	**Date**	**Author**	**Comments**
1	24/01/2011	UCL – Vivian Genaro Motti	Basic structure of the document: table of content, initial content based in the Description of work
2	22/06/2011	UCL – Vivian Genaro Motti	More detailed description of the contents and review of the structure of the document
3	08/07/2011	UCL – Vivian Genaro Motti	Insert diagrams: Use Case and Class, and initial description
4	15/07/2011	UCL – Vivian Genaro Motti	Content
5	02/08/2011	UCL – Vivian Genaro Motti	Updates diagram and content
6	15/08/2011	UCL – Vivian Genaro Motti	Final review 1

7	20/08/2011	UCL –Vivian Genaro Motti	Final Review 2
8	26/08/2011	UCL – Vivian Genaro Motti	Final draft

Executive Summary

The main goal of this deliverable is to present reference models that support developers in the implementation of applications that perform context-aware adaptation. The models formalize the definition of concepts that are relevant for the adaptation process, such as: the context information, adaptation techniques, methods, rules, as well as the strategies that are used to present adaptation.

The elaboration of UML diagrams provides a graphical view of adaptation concepts from different perspectives, for example from the user perspective, the tasks that can be performed, and from the system perspective, the states of execution. The diagrams also provide a unified view of Serenoa and its essential concepts regarding the adaptation process.

In this first release the models are defined in a high abstraction level, generic enough to allow them to be used in a wide range of domains for adaptive and adaptable applications and to accommodate future changes along the evolution of the project. The second and third releases of this deliverable will present the models specified in more details, according to the evolution of the architecture of Serenoa and focusing on application scenarios.

Table of Contents

1 Introduction

1.1 Objectives

This deliverable presents and describes the reference models that formalize the definition of essential knowledge for implementing and performing the adaption process. The modeling phase is an iterative process; for this first release we focus on 5 diagrams types: Use Cases, Sequence, Class, Statechart and a Meta-model. The definitions concern a high abstraction level, generic enough to comprehend all adaptation types (i.e. adaptivity and adaptability) and allowing further specializations that will be reported in the following releases (2 and 3).

1.2 Audience

The target audience consists of researchers and practitioners with interest in modelling the adaptation process.

1.3 Related documents

- The Deliverable 1.2.1 - Architectural Specifications defines the entities used to structure the life lines of the Sequence Diagrams
- The Deliverable 2.1.1 - CARF and CADS contains the catalog of adaptation techniques, dimensions and concepts that aid the composition of the Reference Models (in particular the Use Cases)
- The Deliverable 2.2.1 - CARFO (R1) is related with and complements this document once both documents formally represent the concepts of adaptation, mainly the ones previously and partially defined by the CARF and CADS
- The Deliverable 4.2.1 - Algorithm Library for AAL details and complements the flows (basic and alternative) presented in the descriptions of the Use Case diagrams

1.4 Organization of this document

Chapter 1 presents the objectives, audience and related documents with this deliverable. In Chapter 2, the description of work is presented and illustrated with examples. Chapter 3 describes background information. Chapter 4 specifies the Reference Models. Chapter 5 presents final remarks and concludes this deliverable. The Annex Sections report the descriptions of the Use Cases.

2 Description of Work

2.1 Motivation

The creation of models aims at structuring, organizing and formalizing the relevant information that supports the development of adaptive and adaptable systems.

The motivation to define meta-models is the need of an abstraction that defines rules, constraints, and theories for context-aware adaptation.

2.2 Goal

The definition of reference models intends to provide support for developers and designers to implement the adaptation process. The main goal is to create models that formally define the knowledge about adaptation in order to help the development of adaptive and adaptable applications. The models help the development of these applications by formally defining the knowledge about the concepts, properties and relationships involved with adaptation. In addition to this, the adoption of UML helps us to [Booch et al., 1998]:

- Visualize the system to be
- Specify the structure and behavior of the system
- Define templates to guide the implementation of the system
- Document important decisions

By modeling the related concepts, it is possible to discuss design decisions at an early stage of the development life cycle and consequently to identify potential critical points that may appear during it. This approach also intends to optimize the development phase.

2.3 Description

Task 3.1 Reference Models for CAA of SFEs

The goal of this task is the creation of a set of Reference Models (RMs) that will enable the creation of context-aware SFEs in accordance with the CARF produced in Task 2.1.

The RMs will be concerned with the conceptualization and formal representation of:

- The enriched context of use (Environment, Platform, User, UI Services, Situations, …).
- The different layers of abstraction that are found in the development of UIs (Task and Domain, Abstract and Concrete User Interface).

The RMs will be specified primarily by means of MDA standards (UML 2 and MOF), although other representations (e.g., OWL2) might be provided if considered appropriate during the development of the project.

3 Background Information

This section presents the material, methods and context in which the Reference Models were elaborated. The section starts with the description and discussion about the languages and technologies that were considered. Then we present the concepts used to compose the Reference Models, namely the CARF, the Context Information that are relevant for the Serenoa's context, and the definition of UI models and their transformations.

3.1 Reference Models

There is a set of different manners to model information. The context, for instance, can be modelled with: mark-up languages, object oriented languages, key-value or ontologies [Strang and Linnhoff-Popien, 2004]. In the domain of this project we decided to adopt UML because it is a well-defined modelling language, it is an OMG standard, and thus supported by many different environments.

Although UML has certain limitations in comparison with OWL, these two languages have their own strengths. Therefore our goal is to use them as complementary approaches, defining the models using UML and MOF and then refining them with OWL, by adding for instance constraints that allow reasoning for adaptation.

The models to formalize the concepts related to adaptation were created using UML2 and MOF[1]. The UML models include: Use Case Diagrams, Class Diagrams, Sequence Diagrams and Statechart Diagrams. The MOF was used to generate the meta-model for adaptation. The models were generated with UML2 Tools[2], a set of GMF-based editors to create, edit and visualize UML models. ArgoUML[3] was also used to support this task.

The diagrams provide different perspectives of a system. A Use Case Diagram defines who are the actors that can interact with the system, and also the tasks that they are able to perform. Class Diagrams define the classes and objects of a system, as well as their properties, methods and relationships. A Sequence Diagram defines entities of the system, their communication flows and the messages exchanged. A Statechart Diagram defines the states that are essential to achieve the interaction goal. These diagrams are explained in more details and illustrated in the Section 4 - Specification of the Reference Models

3.2 CARF

The CARF, previously defined and presented in the Deliverable 2.1.1, consists in a graphical representation of adaptation concepts concerning what can be adapted (i.e., which resources), how these resources can be adapted (i.e. adaptation techniques), when (e.g., run time, design time), to what (i.e., aiming which goal) and where (e.g. client, server), who adapts (e.g. the end user), and why to adapt (i.e. aiming software qualities, such as performance).

The concepts defined previously by the CARF provided a base to define the Reference Models. These concepts gathered are being continuously refined during the project. For instance, the *how* branch in CARF lists adaptation techniques, which are also used to define the possible use cases of an application that performs adaptation. For the use cases descriptions, not only more adaptation techniques were considered, but also further refinements were added.

Another example is the *what* branch, that according to the definition of adaptation rule provided by the Class Diagram, corresponds to the parameter *resource* that is modified by an adaptation operation (executed as an action in the adaptation rule).

3.3 Enriched Context of Use

The context information was in a first moment modelled using a MindMap diagram. Then, the categories of

[1] Available at http://www.omg.org/mof/, MOF is as standard to specify models
[2] Available at http://www.eclipse.org/modeling/mdt/?project=uml2tools, UML2 Tools support the models development
[3] Available at http://argouml.tigris.org/, ArgoUML is a UML modelling tool

the information gathered were re-used to define the properties of the classes for the Class Diagram during the modelling phase. The context information is mainly defined as a composition of 4 classes: User, Platform, Environment and Context of Use. Each of these classes is specified with a set of properties, the instantiation of these properties compose the context of the user, and fill the parameters of the adaptation rules.

Due to the fact that the set of properties of the classes consists in a long list, categories abstracting them were included in the classes aiming a clear visualization of the diagram. Besides, there are many works reported in the literature that also present models for context information; they are being considered for the Serenoa context. These additional models mainly aid the refinements of the context model defined, this occurs mainly because they are usually specific and detailed models, i.e. they tackle one single dimension of context, or they target one application domain. Some examples include: the Platform Model of the Delivery Context Ontology by W3C [DCO], the Context of Use Model [NEXOF-RA] and the User Models of [Zimmermann et al., 2007] and [Heckmann, 2005].

The context information must be analysed carefully in order to provide an efficient adaptation for the user, because certain contexts have higher priorities than other ones (e.g. usually the user preferences have higher priority than any other context information). These priorities will be defined in the next phases of the development of the project, for instance during the definition of the advanced logic for adaptation, with machine learning algorithms, and also during the definition of the Serenoa Ontology.

A first in-detail view of the context modelling that will be implemented for SERENOA is available at the deliverable D2.2.1 'CARFO (R1)', section 4.

3.4 Abstraction Layers

Implementing adaptation with a model-driven approach implies in considering different abstraction levels to develop an application. This approach not only permits to better structure and organize the project, but also optimizes the development process. Regarding user interfaces, description languages, such as UsiXML (usixml.org) and MARIA [Paternò et al., 2009], consider these abstraction levels and provide support for developers to implement the UI models and also to execute transformations between them.

In the context of Serenoa, the different UI models and their respective abstraction levels were considered. The models are implemented considering the context information and the adaptation techniques that they support, aiming automatic transformations between them. Regarding the Reference Models, we briefly define them below and use these definitions to compose some of the diagrams (like the Statechart Diagram, the Class Diagram and the Sequence Diagram).

Four models of UI in different abstraction levels are considered in the context of Serenoa, they are briefly defined below, from the highest to the lowest abstraction level [Cameleon]:

- **Task and Domain:** This model specifies the goal of the user in a given domain, and defines the logical activities, i.e., a set of tasks (and their respective sub-tasks) that are required to achieve the user's goal. Usually, it has a hierarchical representation stating the temporal relations and the associated attributes. These definitions are platform independent once they are elaborated in a high abstraction level, however there are specific tasks that can also be defined even being specific for a given platform.
- **Abstract:** In this UI model the low fidelity prototype of the user interface is used to translate the initial requirements into a more concrete representation of the user interface elements required to implement the application. It defines presentation units for the UI, independently of interactors or interaction modality. The units support the execution of the tasks, and group sub-tasks. A high level definition of the navigation is specified.
- **Concrete User Interface:** In this user interface model the UI elements, as well as their rendering details, are specified and defined. It corresponds to a platform-specific model, expresses concrete interactors, dependent of modality and platform. It also defines interactors, widgets, layout and navigation details. However, it is still a mock-up that runs only within a particular environment.
- **Final User Interface:** It is the operational UI, i.e. any UI running in a particular computing platform. It is composed by the source code and the running interface.

Thanks to the four abstraction levels, it is possible to establish mappings between instances and objects found at the different levels and to develop transformations that find abstractions or reifications or combination [Limbourg et al., 2004; Vanderdonckt et al., 2004].

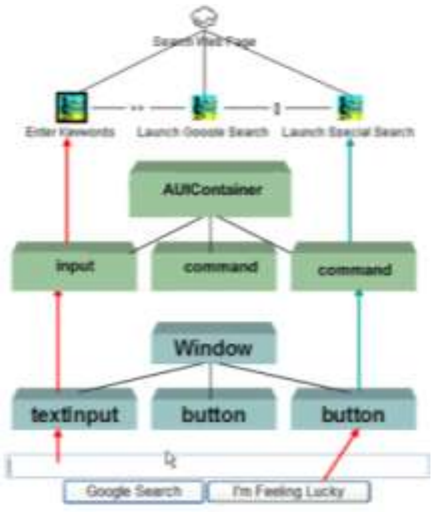


Figure 1. Example of UI models according to the CAMELEON Reference Framework

The Figure 1 exemplifies the four models of UI, the main goal (task and domain model) corresponds to search a web page, which is composed by three sequential tasks, the AUI is composed by an input element and two commands, the CUI is composed by a Window containing one text input and two buttons, as graphically represented in the FUI.

The tasks defined in the Task and Domain are associated with the use case definitions (i.e., adaptation techniques), and the temporal relations of these tasks can be formally defined with a sequence diagram. The transformation between Task and Domain, Abstract UI, Concrete UI and Final UI are formally modelled in the Class Diagram, the Sequence Diagram and the Statechart Diagram.

4 Specification of the Reference Models

4.1 Meta-models

The meta-modelling phase consists in the analysis, construction and development of frames, rules, constraints, models and theories that can be applied in a specific domain. In the context of the Serenoa Project, the elaboration of meta-models aims at abstracting the concepts and relationships that are essential for the implementation of context-aware applications able to execute adaptation processes.

A meta-model diagram is a conceptual diagram, and according to Booch, Rumbaugh and Jacobson (1999), it is an adjusted version of a class diagram. Meta-model diagrams are built as a composition of concepts, generalizations, associations, multiplicity and aggregations.

The Object Management Group[4] (OMG) defined a meta-model architecture composed by four levels. From the lowest to the highest abstraction level: M0 represents the actual instance of a model, i.e., data; M1 represents specific concepts (such as classes and object names) that can be seen as UML models; M2 represents abstract concepts that can be seen as UML meta-models; and M3 represents meta-models that can be seen as a MOF model.

In order to exemplify the concept of meta-modelling, instances of these abstraction levels were elaborated. They are illustrated in the Table 1. This table presents a brief example of the concept of meta-models applied in an adaptation scenario. M3 refers to the abstract concept of Adaptation, which can be defined as the process of modifying some aspect in a given application. M2 defines that to perform the adaptation process, two concepts are necessary: the context information and adaptation rules. In M1 the rules are defined as a set of conditions and actions. These two concepts are associated as: *if condition then action*. A condition has the structure ‘if’, followed by a context information, then a comparator, and an actual value. The action is composed by ‘then’, an adaptation method, and an adaptation strategy. A method can be composed by one or more adaptation techniques, and an adaptation technique consists in an operation (transformation) applied in a given resource. M0 exemplifies an instance of the adaptation rule: *if user is colour blind then daltonize images*, by illustrating the operation (daltonize) applied to an image. M0 is closely related to the definition of Final User Interface (the source code and rendered UI) provided by some User Interface Description Languages (UIDL), such as UsiXML [UsiXML] or MARIA [Paternò et al., 2009].

The relationships between this meta-model and the UML models presented in the next sections are:

- Use Case Diagrams: define a set of techniques for adaptation that can be performed by the users (or the system). In M1 a method is a composition of one or more techniques for adaptation. The techniques are a composition of operation(s) and resource(s); the use cases are also defined according to these concepts, once they transform a given aspect of the application. The CARFO Ontology will provide additional information allowing further inferences about the relationships between adaptation techniques (such as, compatibility level or abstract methods).
- Sequence Diagrams: specify the entities that are responsible to perform the adaptation process and the communication messages exchanged between them. Thus, it refers to a technical view of the adaptation process that supports the implementation of the concepts defined by the meta-models.
- Statechart Diagrams: establish the phases (states) required to perform the entire adaptation process, since the beginning (when the user or the systems wants the adaptation), until the end, when the adaptation is performed, presented, and accepted by the user. This diagram allows a temporal view, related to the navigational steps of the user or the adaptation stages of the process. In this deliverable we opted to define Statechart Diagrams, tackling the views of the user and of the system itself about the adaptation process.
- Class Diagram: defines the adaptation process in terms of classes, objects, their relationships, properties and methods. It defines the key concepts to implement adaptation, refining thus the definitions provided in the meta-model.

The meta-model elaborated for Serenoa focuses in the core of adaptation, defining a vocabulary that is

[4] http://www.omg.org/

specific to support the adaptation process, i.e. handling context information, selecting adaptation rules, and applying the transformations (adaptation techniques) with a specific and compatible strategy.

Table 1. Meta-model Layers

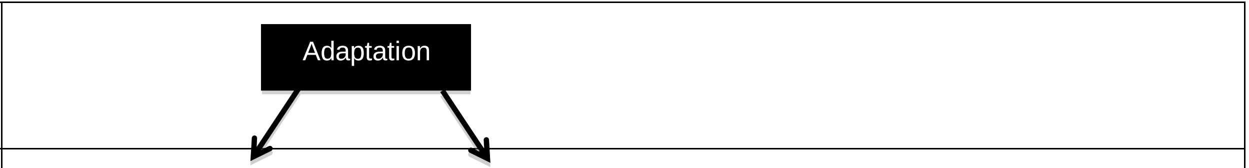
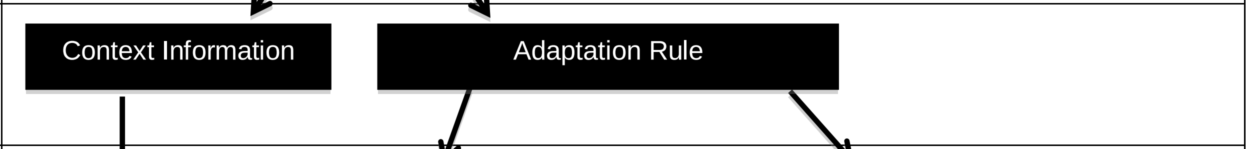
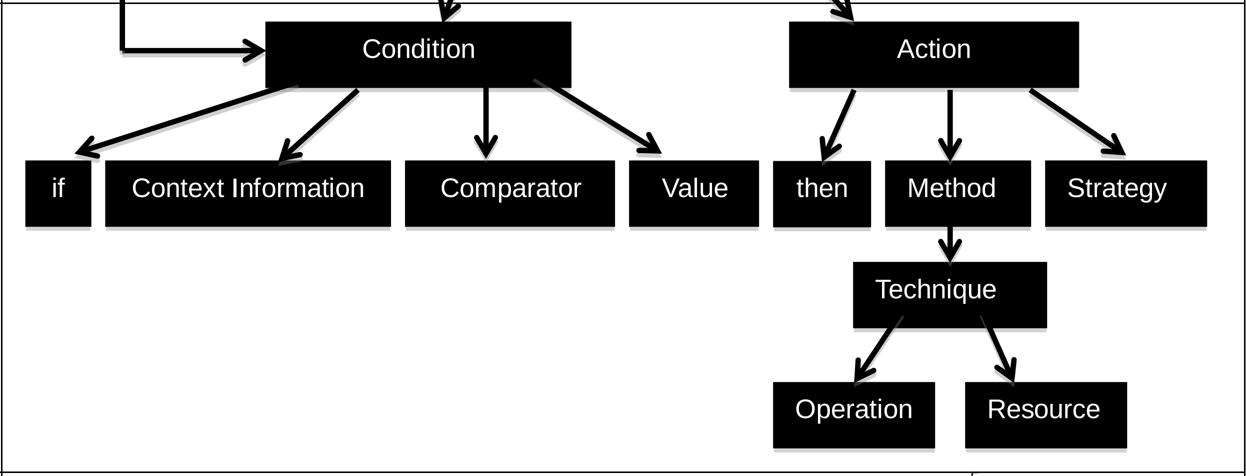
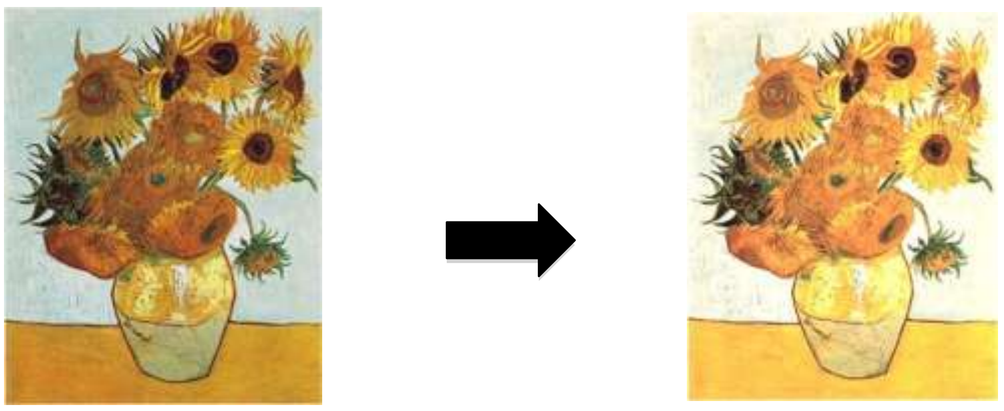
M3	
M2	
M1	 Condition Action if Context Information Comparator Value then Method Strategy Technique Operation Resource
M0	`If user is color blind then daltonize images`[5] 

Table 1 presents an overview of the main concepts of adaptation, organized in four abstraction layers (meta-models, model and instance). Although, this overview presents essential concepts to support the adaptation process, it is not complete, leaving room for further refinements. Thus, in this sense the concepts presented in each layer were refined, mainly with the definitions of the UML models, their descriptions and also with additional contents reported in the related deliverables.

Concerning the example of Table 1, extensions can be considered. Taking as example the specification of the

[5] The source of the images are: http://content4.clipmarks.com/blog_cache/www.vischeck.com/img/B883420B-6F53-47EA-9B39-5423FC9FD649 and http://content1.clipmarks.com/blog_cache/www.vischeck.com/img/ EFB6B39C-EC93-4799-928B-BD18DA1114A9

adaptation rules, the actions can be augmented with two additional concepts, *events* and *justification*, generating thus ECA [Dittrich et al., 1995] and JECA rules [Zongwei et al., 2000]. ECA-rules (event-condition-action rules) are composed by the structure: "when an event occurs, check the condition and if it holds, execute the action". JECA rules (justification-event-condition-action) add also a justification to the event. A justification forms the reasoning context in which the evaluation of the specific JECA rule can be performed. It is frequently used as a disqualifier. For example, the rule is disqualified if J is evaluated to true.

In addition to this, the actions of the adaptation rules can consider classifiers. Classifiers specify the definition of target-resources of an adaptation technique. Assuming that resources are instances of images, the operation (that composes the adaptation techniques) can be applied to *all* images or only to *images that are larger than 100 pixels*.

The Adaptation Techniques can also be further refined with the addition of a parameter, thus in this case the technique would be composed by a triple: operation, resource, and parameter, which could be instantiated as *re-size images in +50%.*

4.2 UML

The Unified Modelling Language (UML[6]) is the most used standard language to support the modelling, specification, construction, visualization and documentation of the concepts involved in the development of an application. In particular the application structure, behaviour, architecture, business process and data structures are covered. UML is an OMG standard currently in version 2. The standardization of UML 2 extended its scope and viability. UML is generic enough to support complex modelling problems across a variety of application domains [UML].

4.2.1 Use Case Diagrams

The Use Case Diagrams describe the tasks (and sub-tasks) that an actor, such as the user, the system or a third party, is allowed to perform while interacting with the system. For the use cases described in this deliverable, the tasks are defined in terms of adaptation techniques that can be implemented and supported by an adaptive or adaptable application. These tasks are previously, and partially, reported in D2.1.1 (R1) CARF and CADS.

Table 2. Use Case Description

Use Case Element	**Description**
Use Case Name	The name of the use case, in a short and clear definition
Use Case Description	A definition of the use case
Primary Actor	Who is the main actor of the use case
Precondition	What are the conditions and requirements to be met before the use case execution
Trigger	What event triggers this use case
Basic Flow	The sequence of steps of the use case when everything is fine (no errors, or exceptions)
Alternate Flows	The most significant alternative flows and exception treatments

In order to detail the use cases, they were defined with: their names, a brief description, the actor ('agent' who triggers the technique), their pre-conditions, the events that trigger them, and their flows (basic and alternative sequence of steps to perform the adaptation technique). The Table 2 presents the concepts used to describe the use cases. Note that some of the use case names are followed by the marks *aka* or *cr*, which refer respectively to *also known as*, or alternative names, and *cross-reference*, or one or more related use cases. These marks were added as an initial effort to associate semantic information to the techniques. Further efforts will be necessary to conclude this process, the goal however is to start to establish

[6] www.uml.org

relationships between the techniques, stating for instance their compatibility degree, aiming to provide future inferences.

The descriptions of the use cases (adaptation techniques) enhance, detail and complement the templates presented at D2.1.1, and also provide a base to compose the Library of Algorithms (to be reported in D4.2.1). The descriptions of the use cases can be found in the Annex A – Use Cases Description for Content.

One or more actors can perform the use cases. The actor, in our context, is abstracted as an *Adapter* and specialized by three roles, depending on who triggers the adaptation process. These specializations are still abstract in order to accommodate roles that are specific for certain domains of application (for instance, the developer, in the context of the Authoring Environment). The Figure 2 illustrates the three possible roles that can trigger the adaptation: the system itself, the user, or a third-party. It is worthy to mention that any combination of these actors is not only possible (since the system supports it), but also recommended in certain cases.

The *Adapter*, illustrated by Figure 2, performs the main use case, which is defined in a general term as *Adapt*, and corresponds to any type of transformation in any aspect of a system. *Adapt* includes three other use cases, namely: *Adapt Content*, *Adapt Presentation* and *Adapt Navigation* (to provide a precise view of the diagrams, we opted to not repeat the presentation of this relationship graphically in all the diagrams).

To provide a clear visualization of the diagrams, the use cases are presented in this deliverable according to their main target resource, i.e., the resource that is subject to the adaptation (content, navigation, and presentation). And once all the use cases belong to the main use case *Adapt*, that has as main and unique actor (the *Adapter)*, the remaining diagrams are illustrated without the graphical icon of the actor.

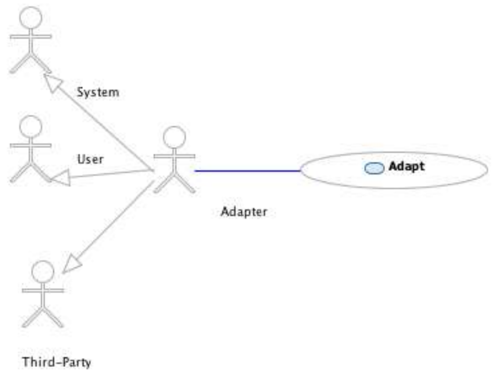


Figure 2. The roles of the actors and the main use case diagram

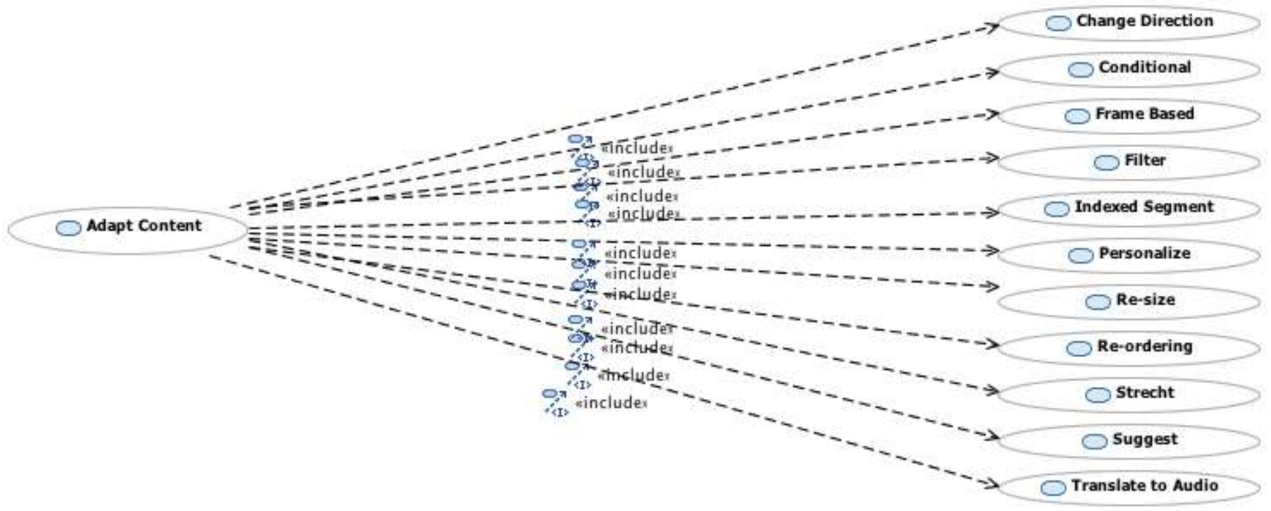


Figure 3. Use Case Diagram for Content Adaptation

The Figure 3 illustrates the use cases for adapting content. This diagram is composed by 11 adaptation techniques that can be applied in a UI to any content regardless of its type. Further information about these techniques can be found in Annex A – Use Cases Description for Content.

Besides these 11 use cases, the use case *Adapt Content* is also composed by other use cases that are refined according to the type of the content they aim to transform, namely: *Audio*, *Image*, *Text*, *Video* and *UI Element*.

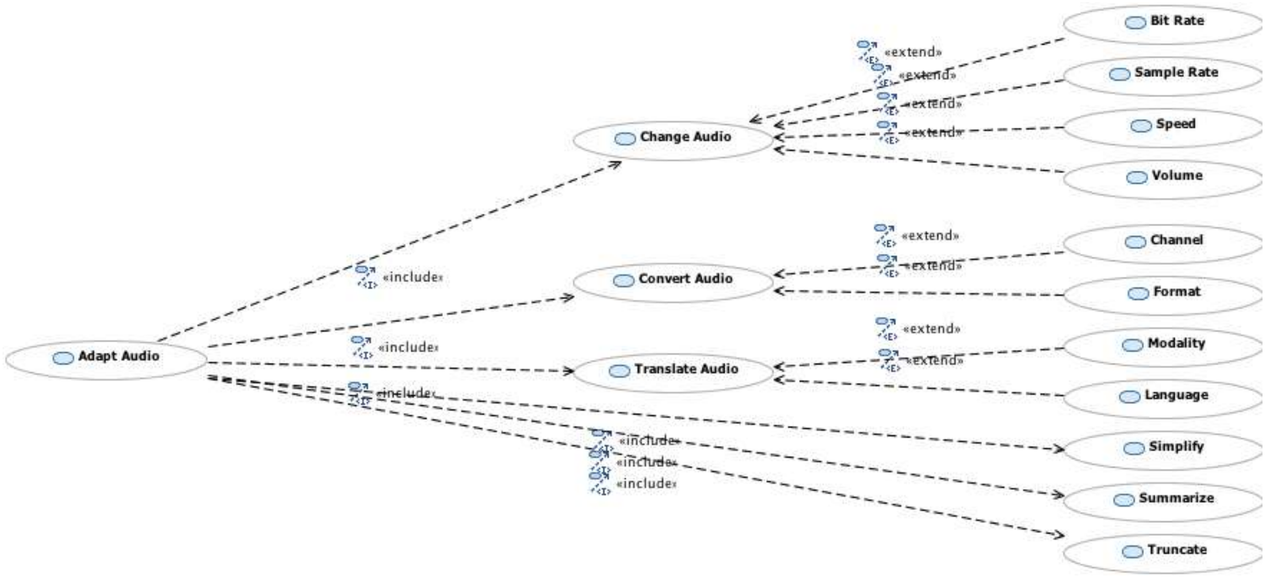


Figure 4. Use Case Diagram for Audio Adaptation

The visualization of the use cases in a graphical and unified view aids to infer potential abstractions about them. Although this phase aimed specifically at modelling the concepts, initial efforts to abstract classes for adaptation techniques were performed. The result of these efforts can be seen in Figure 4, which illustrates the use cases for adapting audio content.

This diagram is composed by 11 adaptation techniques that can be specifically applied to adapt audio contents. In addition to this, three abstract adaptation operations were identified: change, convert and translate. These operations can be applied to specific audio properties, such as the format or the bit rate; that is why they are modelled as extensions. An extension implies a relationship between use cases under certain constraints or conditions. In the use case above, the constraints are defined as specific values defining which property of the audio content will be actually adapted.

Further details about these use cases can be found in Annex B – Use Cases Description for Audio Content.

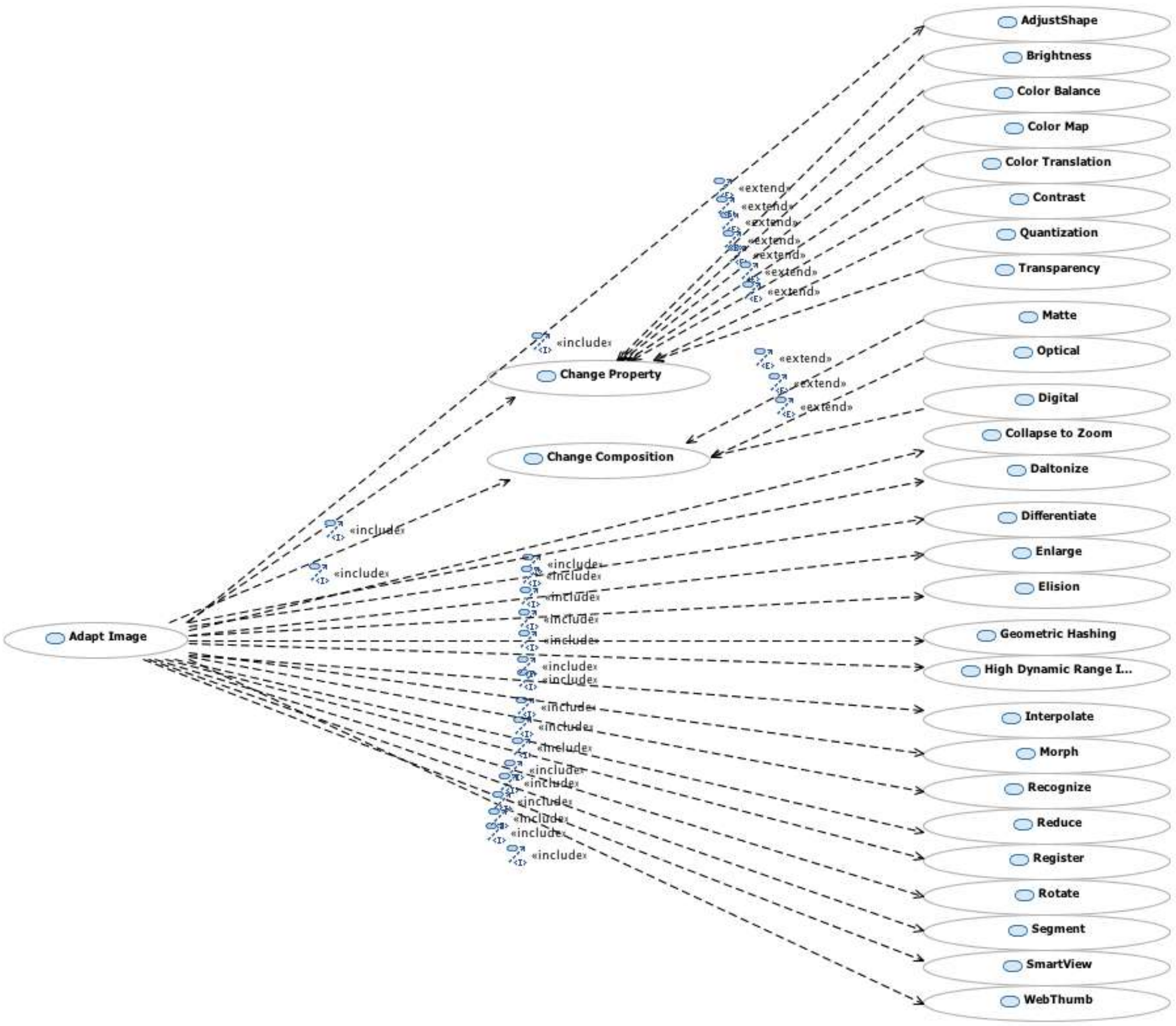


Figure 5. Use Case Diagram for Image Adaptation

The Figure 5 illustrates use cases for adapting image content. This diagram is composed by 27 adaptation techniques to adapt images. Two abstractions were identified in this context: *Change Property* and *Change Composition*, they are extended respectively by 7 and 3 use cases.

Further details about these use cases can be found in Annex C – Use Cases Description for Images.

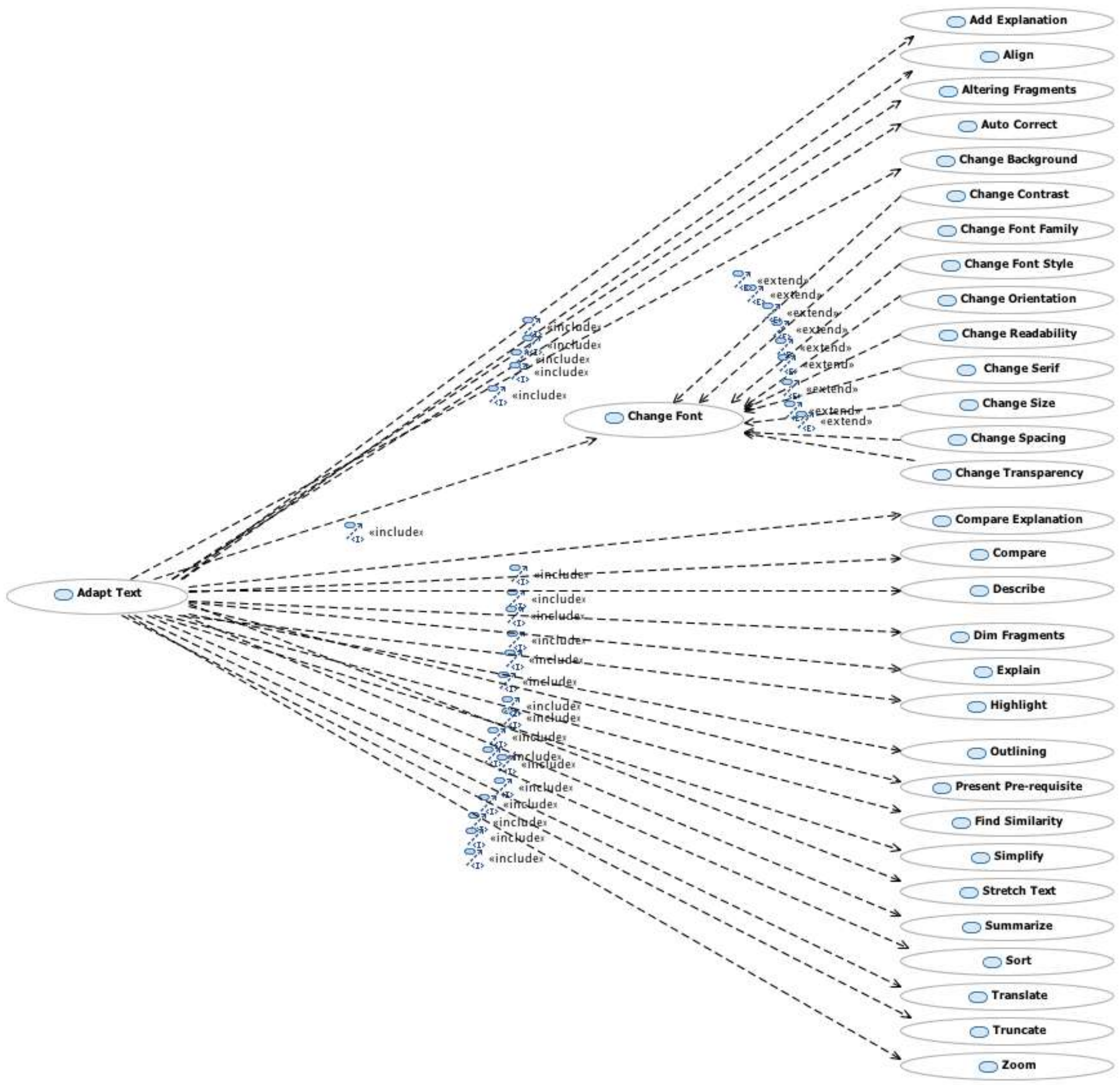


Figure 6. Use Case Diagram for Text Adaptation

The Figure 6 illustrates use cases for adapting text. This diagram is composed by 30 adaptation techniques and one abstract use case, *Change* Font, which is extended by 9 use cases. These use cases can be applied mainly regarding text content.

Further details about these use cases can be found in Annex D – Use Cases Description for Text.

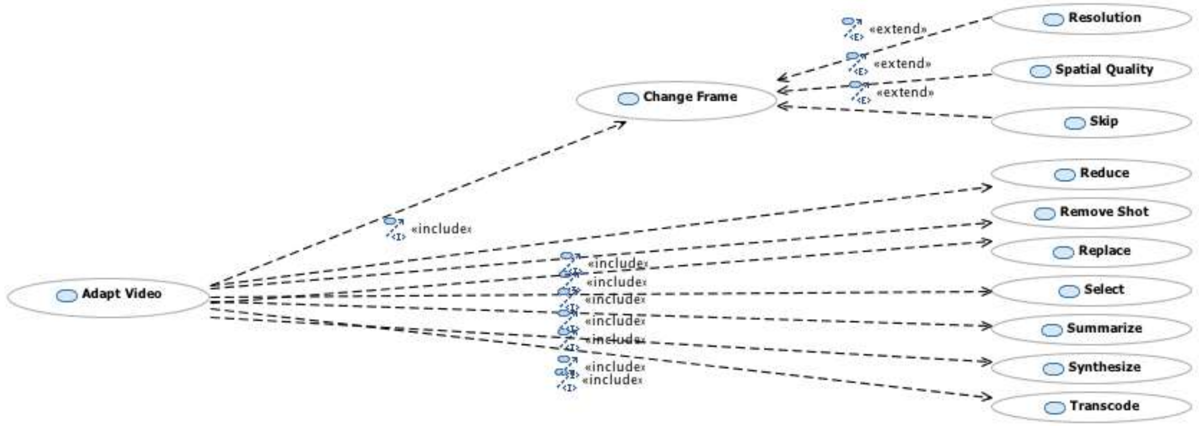


Figure 7. Use Case Diagram for Video Adaptation

The Figure 7 illustrates use cases for adapting videos. This diagram is composed by 10 adaptation techniques and one abstract use case, *Change* Frame, which is extended by 3 use cases. These use cases can be applied mainly regarding video content. Further details about these use cases can be found in Annex E – Use Cases Description for Video.

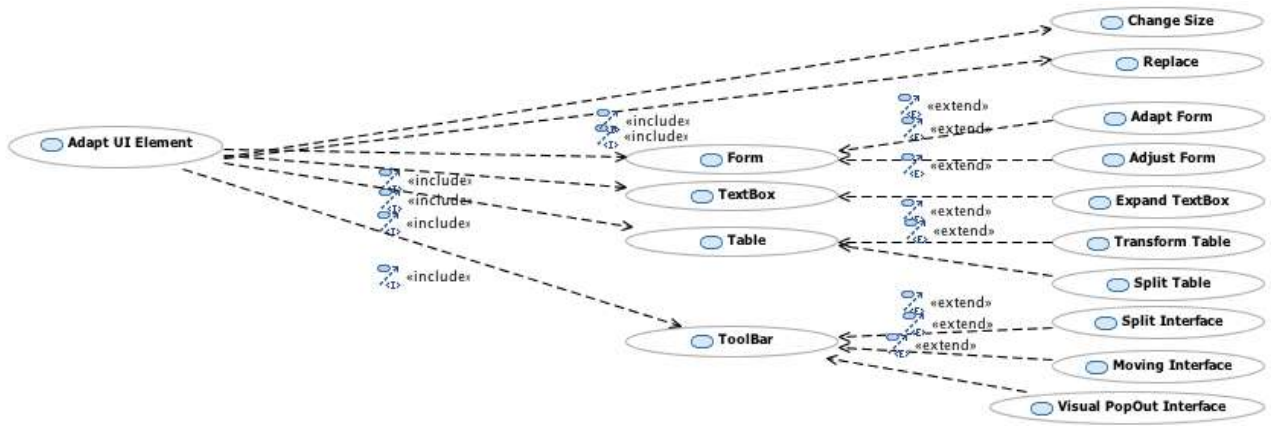


Figure 8. Use Case Diagram for UI Element Adaptation

The Figure 8 illustrates use cases for adapting UI Elements. This diagram is composed also by 10 adaptation techniques but concerning UI Elements, namely Forms, TextBoxes, Tables and ToolBars. The abstract use cases are illustrated by the name of the element, however the action always corresponds to perform adaptation, i.e., Adapt UI Element. Two use cases are generic and can be applied to any element: *Change Size*, and *Replace*. The remaining ones are extensions (8) or abstractions (4). Future versions of this diagram will extend it to include also other UI elements, such as RadioButtons or FilePickers.

Detailed descriptions about these use cases can be found in Annex F – Use Cases Description for UI Element. It is important to remark that in these use cases we have focused on situations in which only GUI interactors are used. In the future it could be possible to add other use cases that feature other non GUI modalities.

The Figure 9 illustrates the use cases for adapting navigation, i.e. the structure and order of links between tasks and contents that the user has available to complete her goal. This diagram is composed by 25 use cases that aim in most of the cases to modify the order in which the links are organized to be presented to the users.

Further explanation about use cases for navigation can be found in Annex H – Use Cases Description for Navigation.

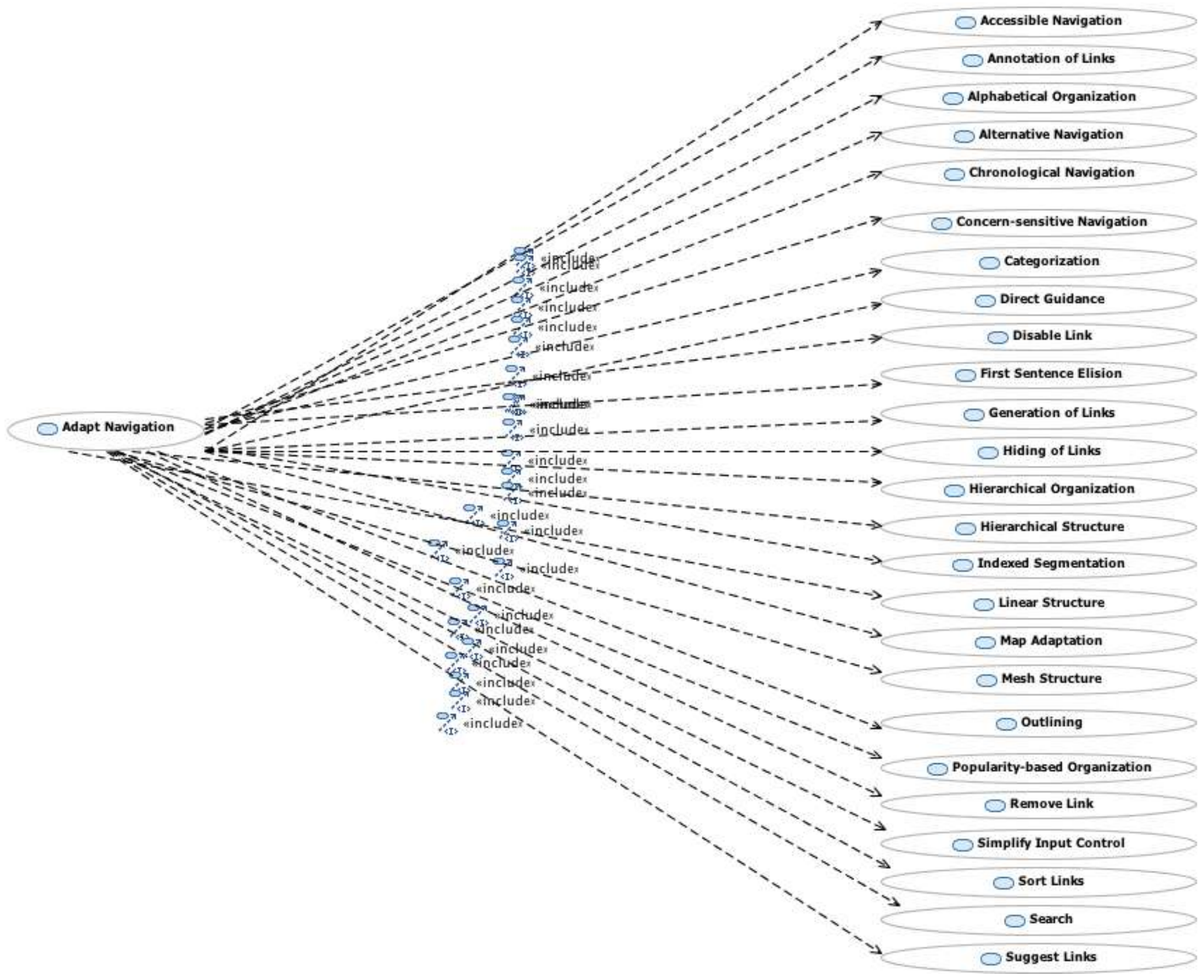


Figure 9. Use Case Diagram for Navigation Adaptation

The Figure 10 (next page) illustrates the use case diagram for adapting the presentation of an application. This diagram is composed by 20 adaptation techniques that can be mainly applied to adapt the presentation. Further details about them can be found in Annex G – Use Cases Description for Presentation.

The use case diagrams were built on top of the adaptation techniques previously specified by the CARF. The elaboration of the models allows us not only to document the use cases of the project, but also facilitates the analysis of them. This analysis aims to abstract the main operations, group and relate similar techniques, elaborate adaptation methods, and so on. The results of these inferences may be used in the development of the ontology of Serenoa (CARFO) and in the implementation of the algorithms for adaptation.

Clearly, not all techniques that were listed must be implemented and available in an adaptive or adaptable application, but certainly the ones that are more relevant for its specific domain must be considered. Additionally, we do not claim here that the use cases and associated diagrams previously discussed cover all the spectrum of adaptation (e.g., the remark that we only covered GUI interactors for the UI Elements adaptation). Further analysis in the future will yield a more comprehensive list.

These use case diagrams define a long list of possible adaptations that can be performed (134 in total) which helps developers and designers to have a complete view and to select techniques for the development life cycle of applications that take adaptation into account.

The information previously reported by the templates of adaptation techniques (D2.1.1) summed with their descriptions in the Annex sections, provide enough information to guide developers and designers in: (i) selecting adaptation techniques that are important in a given domain, (ii) defining which context information must be considered, (iii) specifying adaptation rules, i.e. relating conditions and actions, (iv) choosing possible implementation approaches for the techniques, and (v) deciding appropriate adaptation strategies. The results of our future efforts will also help to guide developers and designers to combine different techniques to execute adaptation methods.

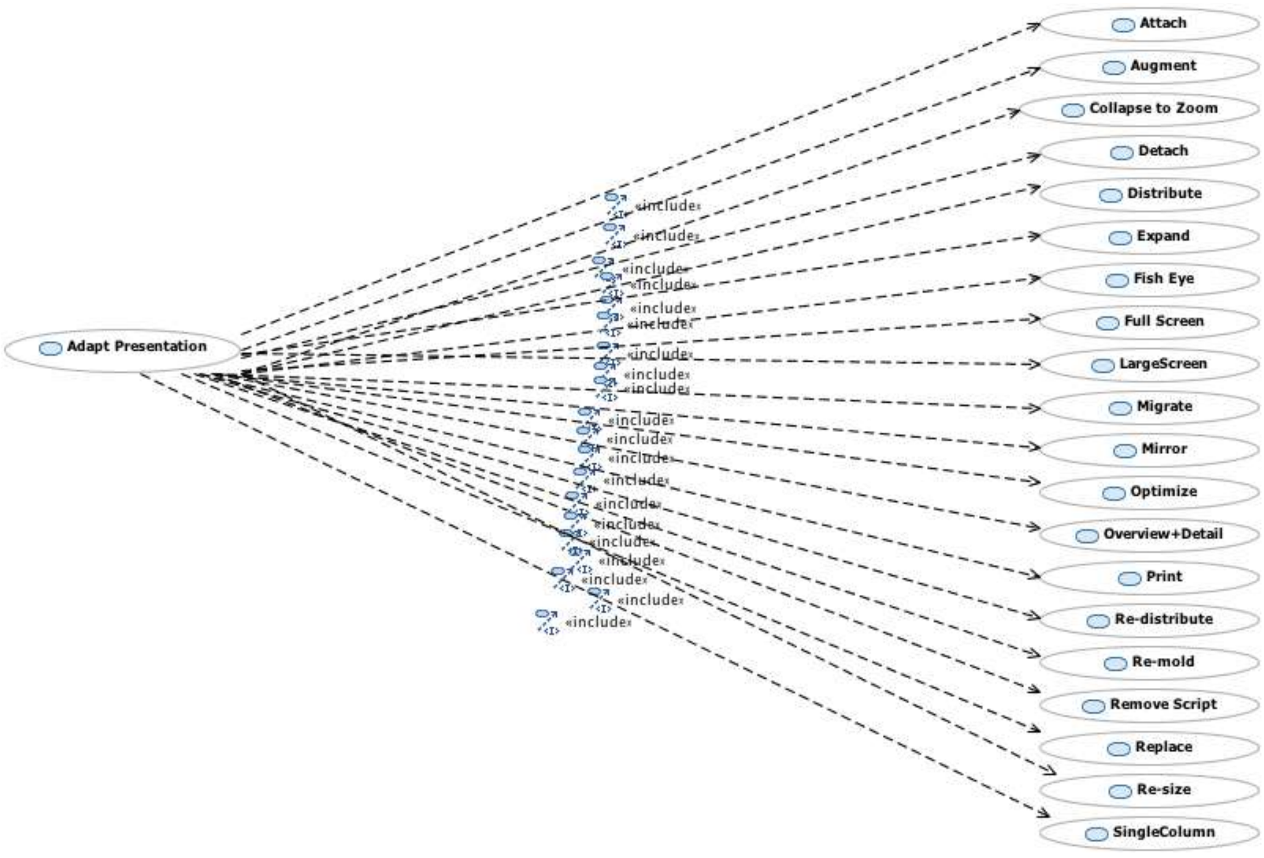


Figure 10. Use Case Diagram for Presentation Adaptation

4.2.2 Class Diagrams

The class diagrams define the main entities of an application, as well as, the properties, methods and relationships that they support.

¡Error! No se encuentra el origen de la referencia. illustrates the class diagram, which is composed by 18 classes. The context class is a composition of 4 other classes, namely user, platform, environment and context of use. The context, once instantiated, i.e., obtained, fills the parameters of an adaptation rule. The adaptation rule follows the JECA structure (explained in Section 4.1) being thus composed by a justification, event, condition and an action. The three first parameters are instantiated by context information. The two first parameters of a rule (justification and event) are optional, and the two last ones (condition and action) are mandatory. An action of a rule triggers an adaptation method, which is composed by techniques and strategies.

An adaptation method must be composed by at least one adaptation technique. Adaptation strategies are optional and they depend on the context information and on the type of adaptation technique applied. The adaptation methods are responsible for generating the UI models, namely Task and Domain, Abstract, Concrete and Final. The methods *transform* permit to convert from one model to another.

Transformations between models can occur in two directions: forward, from a higher to a lower abstraction level, or reverse, from a lower to a higher abstraction level. Forward transformations are called Reification, e.g. when a Task Model is converted into an Abstract User Interface. Reverse transformations are also called

Abstraction, e.g. when an executable code becomes a model. Some UIDLs, like UsiXML, support transformations between models.

The context information can be obtained (by a *get* method) and instantiated (by a *set* method) according to the information extracted by the system. Each context information triggers specific adaptation methods, according to the definitions of the adaptation rules. The methods associate adaptation techniques and strategies, once not always they are compatible.

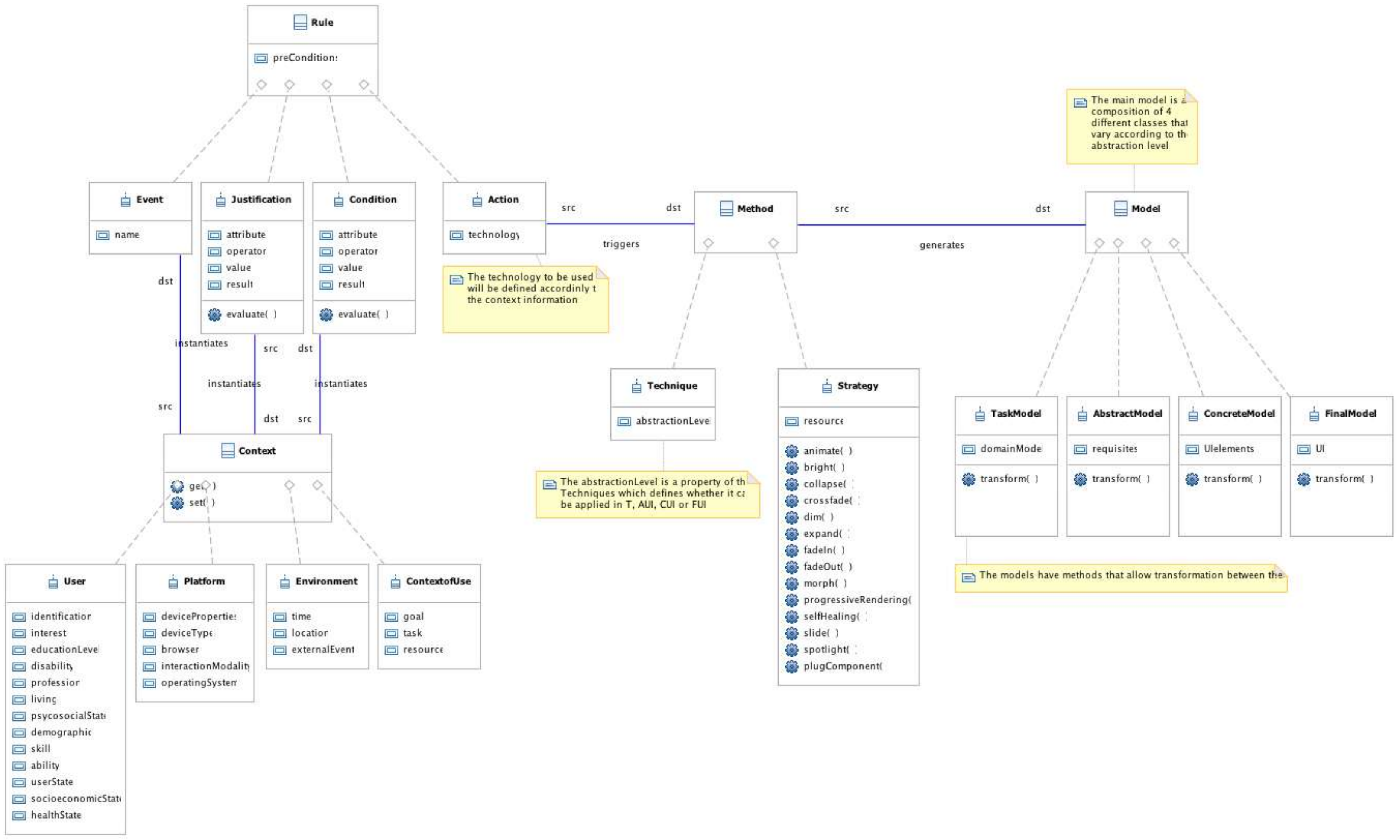


Figure 11. Class Diagram

The machine learning algorithms can support the inference process about which adaptation rules should be selected. The ontology (CARFO) can support the inference process about which adaptation techniques and strategies are compatible, and about the adaptation methods (which techniques compose them, how they are applied). The specifications of these two inference processes will be better defined in the next phases of the Serenoa project.

The classes are defined in a high abstraction level. The properties of the class *User*, for instance, can be expanded in a detailed sub-set of additional properties. Identification is a category of context information about the user, which is composed by name and contact, that are respectively sub-divided into *family, first, last,* and *city, state, country, zip code, email, fax, phone number, and street.* The CARFO ontology defines further details about context information, specifying the properties of the context classes.

In order to detail the classes, objects, relationships, attributes and properties that compose the Class Diagram, we plan to conduct a CRC (Class Responsibility Collaboration) session, a brainstorming technique with cards used to design object-oriented software that supports the definition of the classes and their interactions [Beck and Cunningham, 1989]. This technique focuses on essential classes and defines their responsibilities and relationships. By performing a CRC session we intend to synchronize the different perspectives of the partners regarding the classes, responsibilities and collaborations of the system to be.

4.2.3 Sequence Diagrams

Sequence Diagrams (SD) are considered as Interaction Diagrams, which means that they are dynamic model views. The sequence diagram describes how objects collaborate by specifying: (i) the steps that are required to accomplish the goal, (ii) the communication between the entities of the systems, and (iii) the order in which this communication occurs. Control loops can also be defined in a SD.

A Sequence Diagram is one way to describe interactions in UML2. While lifelines represent the life cycle of entities, arrows between them represent the communication messages exchanged by entities.

The three main elements of Sequence Diagrams are lifelines, messages and execution specification. Messages can represent *operationCall* (synchronous or asynchronous) or *signal* (asynchronous) communication. When a *synchronousCall* is created a *replyMessage* is expected. Specific messages are devoted to the creation of the lifeline (*createMessage*) and its deletion (*deleteMessage*).

The Sequence Diagram elaborated in the context of Serenoa considers the entities defined to compose the architecture of the project. These entities are specified in the deliverable D1.2.1. The Sequence Diagram illustrated by Figure 12 provides an overview of the main entities of the project and the communication flows between them. This diagram is composed by 9 entities, being the *Developer* and the *User* the human entities and the remaining ones subsystems that compose the Serenoa architecture.

The communication is initiated by the developer, who sends information to the Authoring Environment. The messages inform parameters for adaptation rules, and the business logic and functionalities to create or edit an abstract user interface (AUI). Besides, the developer also sends her intents and authored units of a UI.

Then, the user provides her context information to the Context Manager. The context of the user is graphically represented in the upper part of the lifeline because it must be provided in the early execution stages of the system. However, context can be composed by static and dynamic information, and dynamic information, such as heartbeats, change often, and thus must be continuously provided to the system. In addition to this, not only the user can explicitly provide context information, but also devices and sensors embedded in the platform and the environment. This diagram represents only the user as an entity source of context, however an unlimited amount of entities can also contribute in this process.

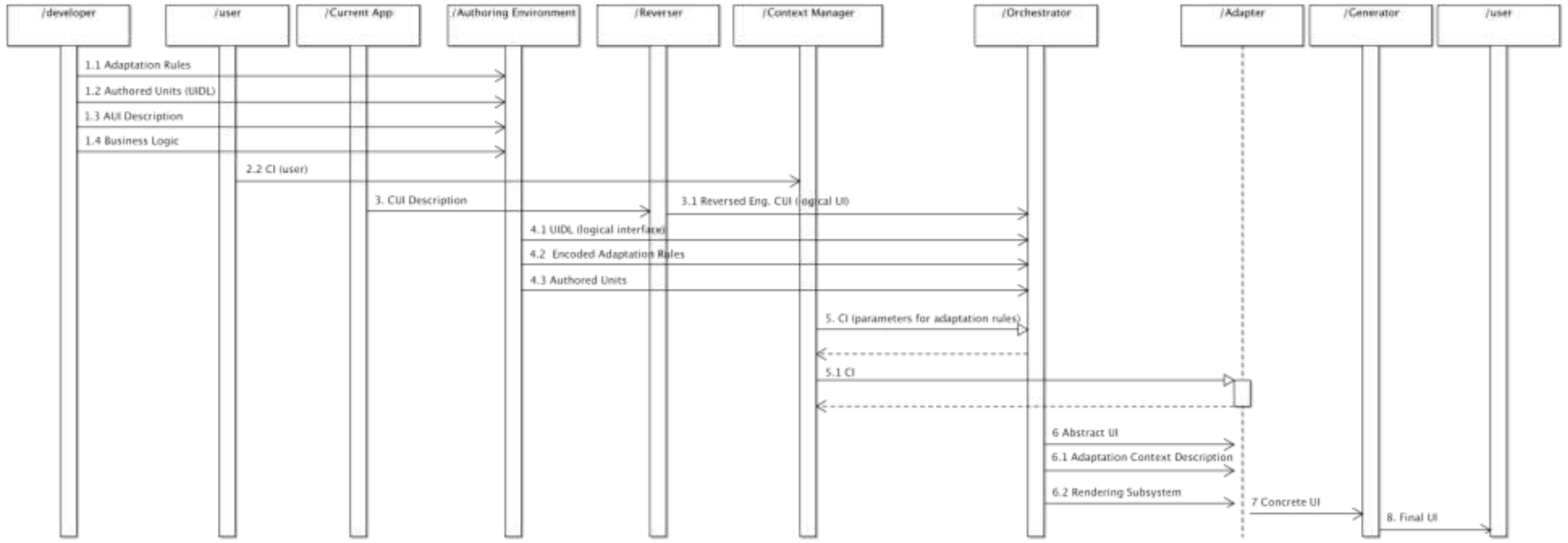


Figure 12. Sequence Diagram

One of the approaches to create an adaptive or adaptable application consists in taking an existing application into account, and generating based on it, a new application version that is adapted. In this sense, the existing application is listed as an entity in the sequence diagram. It provides to the reverser its UI description, which can be for instance, a HTML-based concrete user interface (CUI). The reverser, performs then the reverse engineering of the UI description provided by the existing application, and generates a logical description of the UI, specifying for instance the user interface elements, their layout and their contents. An existing application does not always provide mechanisms, like an API, to access its UI description, in this case the description may be manually extracted or a method must be created to automatically perform this operation.

The Authoring Environment, or Authoring Tool, takes the adaptation rules and authored units, provided respectively by the communication messages 1 and 1.1, codes the adaptation rules, and produces three outcomes: a description of a logical UI, the adaptation rules encoded and the authored units.

The Context Manager based on information received mainly (but not exclusively) from the user, process it to send it to the Orchestrator as values that feed the parameters of the adaptation rules. Once certain types of context information can require a continuous update, a reply message was included in this flow (5). It can be implemented by a RSS feed. The context information from the context manager is also provided to the adapter. The communication flow within the Context Manager is better detailed in another Sequence Diagram (see Figure 13).

The Orchestrator has logical descriptions of UIs, encoded adaptation rules, authored units, and context information. All these data are processed in order to produce the abstract user interface (AUI). The AUI description, together with context information description and the selection of a rendering subsystem are sent to the Adapter.

The Adapter is the entity responsible to transform the AUI in a CUI, taking into account the context information description to apply appropriate transformation rules (according to the rendering subsystems that were previously informed by the Orchestrator).

Once the Adapter generates the CUI, it can be sent to the Generator. The Generator is the entity responsible for transforming the CUI into a Final UI. The Final UI can be then, finally, presented to the user, for instance by being rendered in a browser.

Once the user accesses the adaptation, she may not be satisfied with it. In this case, it must be possible to accept, reject, or evaluate the adaptation. The feedback of the user also counts as context information, which can generate a cycle in the sequence. In this sense, the adaptation must be performed until the user feels satisfied with it. These states are better specified by the Statechart Diagram (see Section 4.2.4).

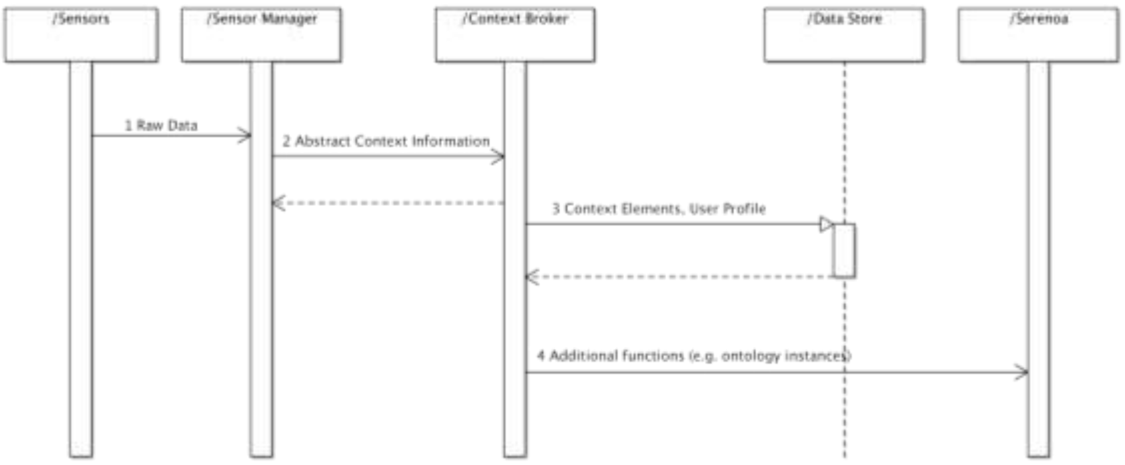

/Sensors
/Sensor Manager
/Context Broker
/Data Store
/Serenoa
1 Raw Data
2 Abstract Context Information
3 Context Elements, User Profile
4 Additional functions (e.g. ontology instances)

Figure 13. Sequence Diagram - Context Manager

The communication flow for the Context Manager involves a set of five entities to be performed. The details about their communication are specified in the Sequence Diagram illustrated by Figure 13. The five entities that compose the Context Manager Sequence Diagram are named: Sensors, Sensor Manager, Context Broker, Data Store and Serenoa (referring to the ‘remaining’ components of the Serenoa architecture).

The Sensors is the entity responsible for collecting context information for the system. Sensors can be of many different types, varying according to the application domain. They collect varied information, such as: weather conditions (e.g., the temperature, humidity level and wind direction/speed), location (e.g., geographic coordinates) or networks available (e.g., Wi-Fi, Bluetooth). The Sensors provide raw data for the Sensor Manager.

Then, the Sensor Manager processes the raw data in order to produce an abstraction of the context information. This abstraction is sent to the Context Broker that extracts relevant information, defining the context elements, such as the user profile. This information is sent to the Data Store, an entity that works as a repository for the context information. The Context Broker can also send additional functions for the rest of Serenoa, for example: ontology instances.

As mentioned before, the context information can be static or dynamic. Dynamic information requires continuous updates. The Sequence Diagrams illustrate the main communication flows of the system, however it is important to notice that dynamic context information will be continuously sensed and updated.

It is important to mention that the quality and the precision of the raw data gathered by the Sensors must be verified and assured by the system. Mechanisms to perform this and also to (pre)process the information will need to be implemented and executed as a pre-condition for using the context information to perform the adaptation process.

The Sequence Diagrams presented above represent the main entities and communication flows for performing adaptation in the context of Serenoa. They represent a high level perspective of the entire process once the concepts used (such as entities and contents and structure of the messages) will be further specified during the next phases of the project and reported in the next releases of this deliverable. The diagrams presented are elaborated in a way to guide developers, designers and researchers in the implementation of the project, by formalizing and documenting relevant concepts. However, it is likely that future changes will need to be accommodated.

4.2.4 Statechart Diagrams

Statechart diagrams define the states that are necessary to perform and conclude a task, i.e., the possible states in which the system (or the user) can be located. In the context of Serenoa two perspectives are considered: the first one corresponds to the possible states of the system itself (execution phases); the second one corresponds to the user perspective, in which the adaptation process is sequentially performed starting with the initial intention of adaptation and finishing when the user is satisfied with the results achieved with the adaptation process.

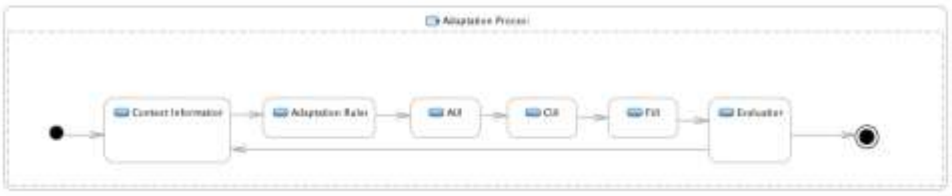


Figure 14. Statechart Diagram: System perspective

The Statechart diagram illustrated in Figure 14 is composed by six states essential to execute adaptation according to the system perspective. The process starts with the gathering of Context Information: when the system senses and extracts relevant information for the adaptation process (or when the user explicitly provides it). Once the context is gathered, it is processed (abstracted, classified, inferred, filtered) to be used to fill parameters of adaptation rules. Then, in the second state, Adaptation Rules are selected, their conditions are instantiated and the transformations specified by their actions, executed. In the third state, the Abstract User Interface (AUI) is generated, followed by the Concrete User Interface (CUI) and the Final User interface (FUI). Finally, in the sixth state, Evaluation, the adaptation process was already accessed by the user via possible adaptation strategies executed to present the FUI, thus the user feedback can be obtained. In case of positive evaluation (acceptance) the adaptation process is concluded, otherwise the user feedback is used to update the context information and the adaptation process re-starts. In case of the user not be satisfied, the context information is updated and the adaptation is performed continuously and iteratively until the user is satisfied. These states are specified in Figure 16.

Another Statechart was implemented to model adaptation according to the user mental model perspective and the seven stages of actions defined by Norman (1986). It is illustrated in Figure 15. The first state is the Goal, when the user aims to achieve something; the second one, called Initiative, occurs when the user thinks which actions in the system are available in an attempt to achieve the goal; the third state is the Specification, when the user defines the set of necessary actions; the fourth state is known as Application, when the user interacts with the system by effectively performing the set of actions; the fifth state is called Transition, when the system reacts to the interaction and changes itself; the sixth state is the Interpretation, when the user perceives and recognizes the changes; and the seventh and last state is the Evaluation, when the user analyses the changes and decides whether the aimed goal was indeed achieved. The evaluation stage can be decomposed in two other states; these are detailed in Figure 16.

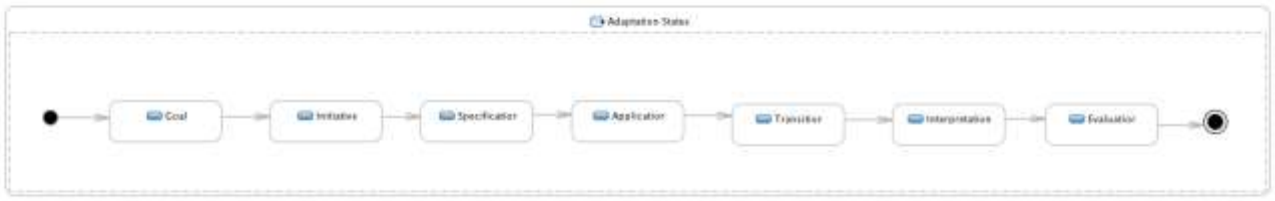


Figure 15. Statechart Diagram according to Norman's 7 stages

According to the evaluation of the user, she may re-start the adaptation process again, either by changing her goal, her initiative, her specification or the application. It depends on the context of use and how the user perceived the changes that occurred in the system. Clearly, if after the evaluation stage, the user is not satisfied with the results obtained, she may re-define its interaction approach to try to reach the original goal. In intuitive systems the goal is quickly reached, however if many attempts are necessary to find the right approach, the user may get frustrated and give up before the goal is achieved. It may happen for two reasons: one occurs when the application cannot perform what the user the wants, the other occurs when the user simply gives up after a given number of attempts.

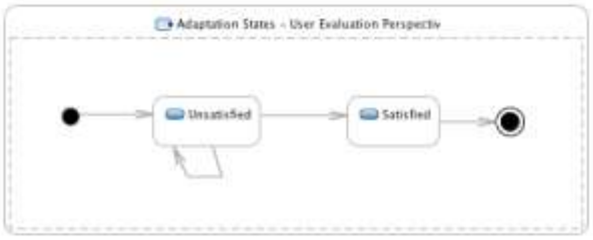


Figure 16. Statechart Diagram: Evaluation Phase

The Figure 16 illustrates the Statechart diagram by the perspective of the user during the evaluation state. Once the adaptation process is performed, the user may feel satisfied or not with its results. If the user is not satisfied, the system must have alternative options to perform adaptation. Logically, in an ideal scenario, the system is able to predict the best approach that suits the user needs, selecting and performing the right adaptation techniques with the right strategy at the first attempt. However, in complex scenarios, involving sets of context information, adaptation techniques and strategies, it may be hard to find out exactly what the user expects from the system at a first moment. Thus, it is necessary to identify whether the user is satisfied with the adaptation performed, and propose alternative options in case of no satisfaction.

All these states can be thought under the light of adaptation, however for adaptable systems the user has the goal, while for adaptive ones the system itself triggers the adaptation process.

4.3 MOF

The MOF (Meta-Object Facility) defined by OMG (Object Management Group) is a formal manner to define meta-models, i.e., models with a higher abstraction level, which can be instantiated according to the adaptation process being performed.

4.3.1 Meta-model for Adaptation

Fuchs et al. (2005) created a meta-model for context information. Their base constructs are: *entity*, *datatype*, *property, quality*, *transformation*, and *value*. They verified the use of the model applying it for an intelligent answering machine application. A meta-model for Context of Use should contain as main classes: *context of use*, *environment*, *user*, *context element*, *entity*, *property*, and *provider*, according to[7] [NEXOF-RA]. Farias et al. (2007) believe that there is no meta-models developed so far that consistently represent context information formalizing properly its syntax and semantics. In this sense, they modelled the development of context-aware mobile applications. A formal meta-model helps to produce valid representations of context that could be consistently used and manipulated throughout the system development, deployment and operation.

Lopez-Jaquero et al. (2008) elaborated a meta-model representing adaptation rules, it is composed by 7 classes associated in the following manner: the *sensor* produces a *contextEvent* that triggers an *adaptationRule*; this rule produces *transformation* and *data*. The transformation can be composed by one or more *transformationRule*. And the adaptation rules are applicable in a given *contextPrecondition.*

As reported above, some efforts have been done as attempts to model context-aware applications. To create the meta-model for context-aware adaptation, first, some related works were analysed. Most of them are specific regarding the domain modelled. Then, this analysis allowed us to identify concepts that are relevant for context-aware applications, mainly regarding the adaptation process, and then finally to generate an initial version of a meta-model diagram. Figure 17 illustrates it. This meta-model was generated according to the other reference models created for the project Serenoa and reported in the previous sections of this deliverable. As adapter, the three main actors defined for the use case diagrams are considered. They initialize the adaptation process, which starts by the selection of adaptation rules. The parameters of the rules follow the JECA structure and are mainly filled by the context information acquired (except the action). The context is a composition of different relevant information for the system, basically composed by elements, properties and qualities, and belonging to the domains of Users, Platform, Environment and Context of Use, as previously mentioned. A condition of a rule can be composed by one or more pre-conditions, and the action of a rule triggers the execution of an adaptation techniques. Techniques that can be structured as a set of: operations, classifiers, resources, one or more properties of these resources, and parameters. One example of instantiation is: ‘diminish all images height in 50%’. The execution of an action generates a UI model, which can be of four different abstraction levels, as defined earlier, from task and domain, to the final user interface, already adapted and presented to the end user.

4.4 OWL

Web Ontology Language (OWL) is the most expressive language for representing and sharing ontologies over the Web. It is designed to be used by applications that need to process information rather than just presenting it. It facilitates the interoperability of web content by providing additional vocabulary and a formal semantics.

The Ontology that refines the definition of the context information for adaptation is currently being elaborated and it will be reported in Deliverable 2.2.1. This ontology will be populated with real examples of use. OWL is able to extend the definitions that UML propose, mainly by allowing developers to add constraints and semantics to the models.

[7] http://forge.morfeo-project.org/wiki_en/index.php/Context_Of_Use_Metamodel#Context_Of_Use

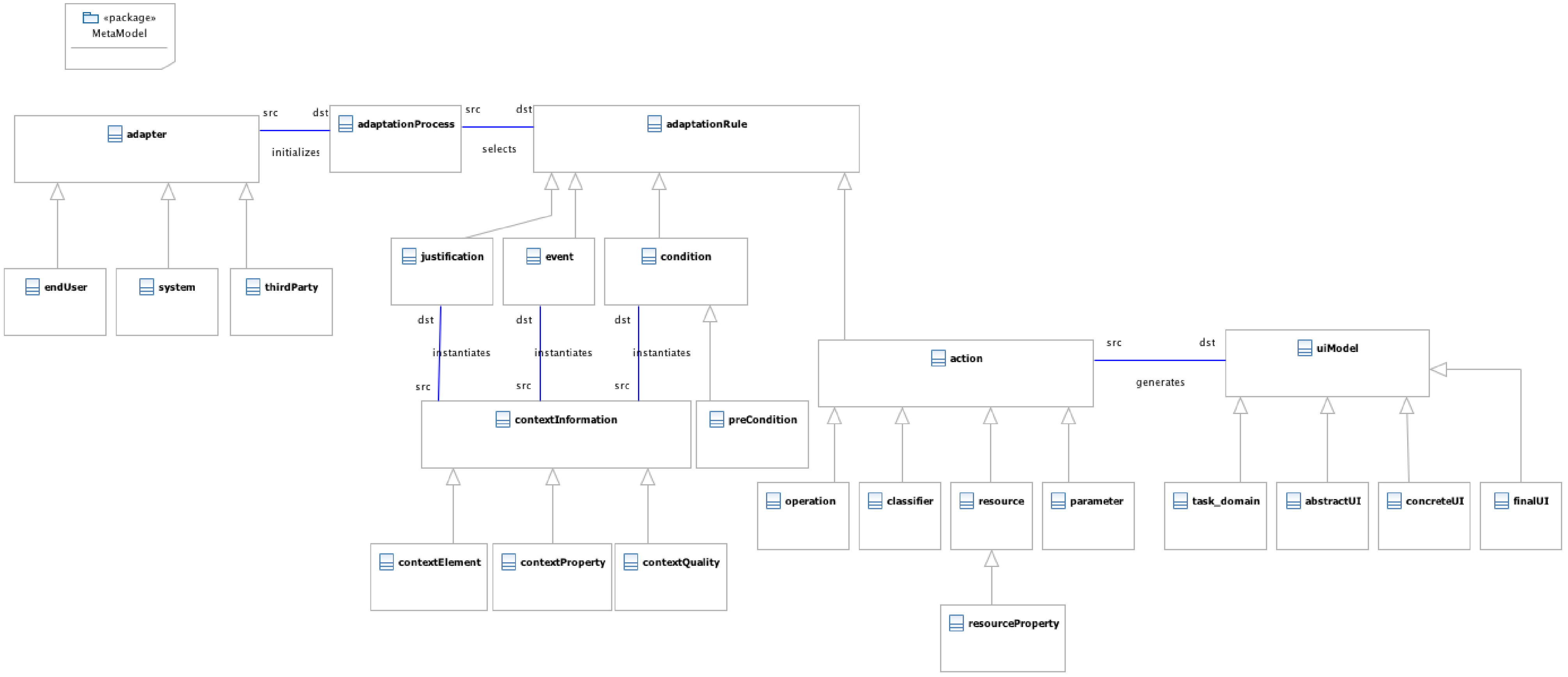


Figure 17. Meta-model for context-aware adaptation

5 Conclusion

The implementation of the UML models intends to support the stakeholders' tasks during the development life cycle of applications. For Serenoa we chose to use UML and MOF as modelling languages in order to formally define the context information, according to definitions previously listed by the CARF, and to model concepts that are relevant for context-aware adaptation.

5.1 Final Remarks

There are many different approaches to model information relevant in the development of a system [Strang and Linnhoff-Popien, 2004], and the adoption of UML models aims at a standard approach. UML is more mature than OWL, and well known by the community of software engineers, which makes the developed models largely accessible. Clearly, the definition of the CARFO Ontology will be complementary to the UML models defined, enhancing them in specific aspects, such as the definition of constraints and additional semantic information.

The models elaborated cover a wide range of aspects of adaptation from different perspectives. While the class diagrams and meta-models cover the classes, objects, properties, methods and relationships that define the concepts of the adaptation process, the Sequence Diagram and Statechart Diagram define dynamic aspects of the system, such as communication messages, execution states of the system and the order of events. The Use Case Diagrams, on the other hand, focus on adaptation techniques that can be performed by the user, or the system itself.

The models elaborated are complementary and aid developers to be aware of relevant information to implement context-aware adaptive or adaptable systems. The graphical visualization of this information helps developers to have a unified perspective of the adaptation process and consequently to start the inference process in order to abstract methods, specialize techniques and also define associations among them.

At this stage of the project, the models are better defined in a high level perspective, in order to accommodate different application domains and to accommodate potential future changes.

5.2 Future Work

The next efforts will consist in detailing and specifying the models in lower abstraction levels. It is likely that some changes will be necessary to accommodate future decisions along the evolution of the project.

The models were defined in a high abstraction level to allow extensions for specific application domains.

The concepts defined by the models will be re-used to compose other information about the project. The use cases descriptions, for instance, provide information that will be refined to compose the library of algorithms. And the concepts modelled by the Class Diagram and Meta-models, provide information that aids the implementation of the Ontology (CARFO) of Serenoa.

References

[Beck and Cunningham, 1989] K. Beck and W. Cunningham. 1989. A laboratory for teaching object oriented thinking. In Conference proceedings on Object-oriented programming systems, languages and applications (OOPSLA '89). ACM, New York, NY, USA, 1-6. DOI=10.1145/74877.74879 http://doi.acm.org/10.1145/74877.74879

[Booch et al., 1998] Grady Booch, James Rumbaugh, Ivar Jacobson. 1998. Unified Modeling Language User Guide, the (1st Edition) (Addison-Wesley Object Technology Series). Addison-Wesley Professional.

[Cameleon] CAMELEON (Context Aware Modelling for Enabling and Leveraging Effective interactiON) Project (FP5-IST4-2000-30104), http://giove.isti.cnr.it/projects/cameleon.html

[DCO] 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/

[Dittrich et al., 1995] Dittrich, K.; Gatziu, S.; and Geppert, A. 1996. The active database management system manifesto: a rulebase of ADBMS features. SIGMOD Rec. 25, 3 (September 1996), 40-49. DOI=10.1145/234889.234896 http://doi.acm.org/10.1145/234889.234896

[Farias et al., 2007] Clever R. G. de Farias, Marcos M. Leite, Camilo Z. Calvi, Rodrigo M. Pessoa, and Jose G. Pereira Filho. 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

[Fuchs et al., 2005] Florian Fuchs, Iris Hochstatter, Michael Krause, and Michael Berger. 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

[Heckmann, 2005] Heckmann, D., 2005. Ubiquitous User Modeling. PhD Thesis, Saarland University, Saarbrücken, Germany. Henderson, A. and Kyng, M., 1992. There's no Place like Home: Continuing Design in Use. In: J. Greenbaum

[Limbourg et al., 2004] Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L., Florins, M., UsiXML: A User Interface Description Language Supporting Multiple Levels of Independence, in Proceedings of Workshop on Device Independent Web Engineering DIWE'04(Munich, 26-27 July 2004), M. Lauff (Ed.), Munich, 2004.

[Lopez-Jaquero et al., 2008] López-Jaquero, V., Montero, F., and Real, F. Designing user interface adaptation rules with T: XML. In Proceedings of IUI. 2009, 383-388.

[NEXOF-RA] NEXOF-RA Project, http://www.nexof-ra.eu

[Paternò et al., 2009] Paternò F., Santoro C., Spano L.D., MARIA: A Universal Language for Service-Oriented Applications in Ubiquitous Environments, ACM Transactions on Computer-Human Interaction, Vol.16, N.4, November 2009, pp.19:1-19:30.

[Strang and Linnhoff-Popien, 2004] Strang, T. and Linnhoff-Popien, C. A Context Modeling Survey. In Proc. Workshop on Advanced Context Modelling, Reasoning and Management 2004.

[Tutorial on Sequence Diagrams in Papyrus MDT] Available at: http://www.eclipse.org/modeling/mdt/papyrus/usersTutorials/resources/PapyrusTutorial_OnSequenceDiagrams_v0.1_d2010100.pdf Access: 01 Aug 2011

[UML] UML Specification. Available at: www.uml.org

[UsiXML] UsiXML Specification. Available at: usixml.org

[Vanderdonckt et al., 2004] Vanderdonckt, J., Limbourg, Q., Michotte, B., Bouillon, L., Trevisan, D., Florins, M., UsiXML: a User Interface Description Language for Specifying Multimodal User Interfaces, in Proc. of W3C Workshop on Multimodal Interaction WMI'2004 (Sophia Antipolis, 19-20 July 2004).

[Vanderdonckt et al., 2005] Vanderdonckt, J., Grolaux, D., Van Roy, P., Limbourg, Q., Macq, B., Michel, 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, 20-22 July 2005), International Society for Computers and their Applications, Toronto, 2005, pp. 207-214. (FP6 Similar NoE)

[Zimmermann et al., 2007] Andreas Zimmermann, Marcus Specht, and Andreas Lorenz. 2005. Personalization and Context Management. User Modeling and User-Adapted Interaction 15, 3-4 (August 2005), 275-302. DOI=10.1007/s11257-005-1092-2 http://dx.doi.org/10.1007/s11257-005-1092-2

[Zongwei et al., 2000] Zongwei Luo, Amit Sheth, Krys Kochut, and John Miller. 2000. Exception Handling in Workflow Systems. Applied Intelligence 13, 2 (August 2000), 125-147. DOI=10.1023/A:1008388412284 http://dx.doi.org/10.1023/A:1008388412284

Acknowledgements

- TELEFÓNICA INVESTIGACIÓN Y DESARROLLO, http://www.tid.es
- UNIVERSITE CATHOLIQUE DE LOUVAIN, http://www.uclouvain.be
- ISTI, http://giove.isti.cnr.it
- SAP AG, http://www.sap.com
- GEIE ERCIM, http://www.ercim.eu
- W4, http://w4global.com
- FUNDACION CTIC http://www.fundacionctic.org

Glossary

- AKA: also known as
- ArgoUML: tool to support the creation, edition and visualization of UML models
- AUI: Abstract User Interface
- Cameleon RF: A Reference Framework defining UI models of four abstraction levels
- CADS: Context-aware Design Space for Adaptation
- CARF: Context-aware Reference Framework for Adaptation
- CD: Class Diagrams
- CR: cross-reference
- CUI: Concrete User Interface
- FUI: Final User Interface
- GMF: Graphical Model Framework
- MARIA: User Interface Description Languages, XML-based
- MDA: Model Driven Architecture
- MDE: Model Driven Engineering
- MM: Meta-model
- MOF: Meta-Object Facility
- OMG: Object Management Group
- QVT: Query-View-Transformation
- SD: Sequence Diagram
- UCD: Use Case Diagram
- UI: User Interface
- UML: Unified Modelling Language
- UsiXML: User Interface Description Language, XML-based

Further definitions can be retrieved at: http://serenoa.morfeo-project.org/wiki/index.php/CommonGlossary

Annex A – Use Cases Description for Content

A template describing the structure of the use cases definition (based on the definition at http://www.gatherspace.com/static/use_case_example.html)

Use Case Name	Change Direction (aka Change Orientation, Rotate)
Use Case Description	Modify the orientation of the content according to the user context
Primary Actor	User / System
Precondition	There is content (text, UI elements, …) composing the user interface, the user has a different orientation scheme (preference, screen dimension, device position, culture)
Trigger	The user has a certain orientation scheme, which is different from the current one (e.g. top to bottom, to bottom to top); the user changed the orientation of the device screen (detected by accelerometer for instance).
Basic Flow	The user interface elements are selected, and the layout is re-defined according to the orientation scheme of the user
Alternate Flows	

Use Case Name	Conditional
Use Case Description	Modify the content order or sequence of presentation according to the user context (for instance pre-requisites)
Primary Actor	User / System
Precondition	There is content (text, UI elements, …) composing the user interface, there is a logic defining the order of the content, or defining the conditions in which that content may or not be exhibited (i.e. restrictions)
Trigger	The accomplishment of a task; the value filled as input (for instance in a form field).
Basic Flow	First the resources are classified according to the conditions under which they will be presented, then the logic to trigger and present it is implemented and performed
Alternate Flows	

Use Case Name	Frame-based
Use Case Description	Provide alternative content for the user that can be helpful to support the original goal (e.g. examples, links, several explanations)
Primary Actor	User / System
Precondition	The user task can be supported by additional content; there are alternatives available to be presented.
Trigger	The user has a complex task to be performed; the user is novice; the user is having troubles in accomplishing one task
Basic Flow	The alternative content is prepared to support the user task and it is provided a way to access it
Alternate Flows	

Use Case Name	Filter
Use Case Description	Given a pre-defined criteria, such as the age of the user is under 18, the content is filtered and only part of it is presented

Primary Actor	User / System
Precondition	There is a pre-defined criteria, and a restriction about the content to be exhibited (for instance a property of the content, such as size)
Trigger	The context of the user (e.g. user preferences, profile)
Basic Flow	The content that matches the selection criteria is selected, filtered, and then presented (rendered) in the final user interface
Alternate Flows	

Use Case Name	**Indexed Segment**
Use Case Description	This transform takes an input page, segments the content into sub-pages by allocating some number of items to each and builds and prepends an index page to the collection.
Primary Actor	User / System
Precondition	There is a set of content to be displayed in a screen that is not big enough
Trigger	The context of use: a small screen device such as a PDA, pager or a mobile phone
Basic Flow	The content is analyzed, extracted, and the index page is constructed by copying a section header or first sentence from each element output, concatenating them onto the index page and creating a hypertext link from each to the appropriate sub-page. The index page itself may need to be segmented. 'Next' and 'Previous' navigation links between sequential sub-pages are also added for navigational convenience.
Alternate Flows	

Use Case Name	**Personalize**
Use Case Description	Modify the content to be exhibited according to the user preferences
Primary Actor	User / System
Precondition	There is content and a user model defining the preferences for the presentation of the content (such as color, background or layout)
Trigger	The user preferences and the presence of the resource to be personalized (such as background color, or image)
Basic Flow	The user (manually or automatically) has his or her preferences defined, the content is adapted accordingly, and presented according to his or her preferences
Alternate Flows	

Use Case Name	**Re-size**
Use Case Description	Modify the size of the content according to the user context (such as preferences, disabilities or constraints of the device)
Primary Actor	User / System
Precondition	There is content (text, UI elements, …) composing the user interface
Trigger	The user has a certain characteristic defined (for instance reduced visual ability)
Basic Flow	The user interface elements are selected, and their sizes are re-defined according to the context (larger or smaller)
Alternate Flows	Besides the re-sizing of the content, it may be also necessary to adjust the

	layout of the interface

Use Case Name	**Re-ordering**
Use Case Description	Modify the order of the content according to the user context (such as preferences, disabilities)
Primary Actor	User / System
Precondition	There is content (text, UI elements, …) composing the user interface
Trigger	The user has a certain characteristic defined (for instance reduced visual ability)
Basic Flow	The user interface elements are selected, and their sizes are re-defined according to the context (larger or smaller)
Alternate Flows	Besides the re-sizing of the content, it may be also necessary to adjust the layout of the interface

Use Case Name	**Stretch**
Use Case Description	From the user model and the domain model, the system determines which fragment should be displayed. For each fragment of information there is a short visible place holder. Stretched fragments must be shown and the others shrunk; this decision determines the initial presentation of the fragment, once the content may be stretched or shrunk through mouse clicks for instance
Primary Actor	User / System
Precondition	There is content (text, UI elements, …) composing the user interface, and a user and domain models defining the contents that should be stretched or shrunk
Trigger	A change in the platform type, reducing the screen dimension may trigger this technique
Basic Flow	The user interface elements are selected, and the layout is re-defined
Alternate Flows	

Use Case Name	**Suggest**
Use Case Description	Given the context of the user, the system provides also alternative options, such as suggesting additional features, or modalities
Primary Actor	User / System
Precondition	There is content (text, UI elements, …) composing the user interface and also a set of related options that can be suggested to the user during the interaction (or after a task was concluded)
Trigger	This adaptation can be triggered by the accomplishment of a task (for instance, once the user finishes reading an article, the system automatically suggests related contents, such as videos, images or texts)
Basic Flow	There is a logic behind associating additional content that is related with the current task of the user, besides there is an event defined to trigger the presentation of the suggestions
Alternate Flows	The suggestions can be provided collaboratively, for instance by users who share the same interests, or are 'contacts' of the end user

Use Case Name	**Translate to Audio**

Use Case Description	Given a content, it is translated to an audio format
Primary Actor	User / System
Precondition	There is content, such as a text, or an image, that can be converted into an audio format
Trigger	This adaptation can be triggered when the user is in an environment that unable the interaction in a visual way, e.g. when she is driving a car, or when the user has a visual impairment; the absence of screen may also trigger this technique
Basic Flow	The text content can be 'simply' rendered with a speech tool, the video content can be presented with its audio part, and an image or animation can be described in terms of alternative text or description in an auditory format
Alternate Flows	

Annex B – Use Cases Description for Audio Content

Use Case Name	**Change Bit Rate**
Use Case Description	The bit rate of the audio content is modified
Primary Actor	User / System
Precondition	There is audio an audio content, and it is possible to change its bit rate
Trigger	The user has a slow connection rate and needs to download audio content
Basic Flow	The content is available, the original bit rate is detected, the aimed one is defined, the file is decoded and then re-encoded at a lower or higher bit rate (e.g. 64 kbps to 44 kbps) (lowering the bit rate makes the file smaller, quality is also lost in this process)
Alternate Flows	

Use Case Name	**Change Sample Rate**
Use Case Description	Given an audio content the sample rate is modified
Primary Actor	User / System
Precondition	There is audio content, the sample rate is known (original and aimed), and it is possible to change the rate
Trigger	The need to have a higher or lower quality content
Basic Flow	Given an audio content, some samples are removed or inserted accordingly to provide a re-sampled content; a method is applied to modify the number of snapshots of audio that were captured and compose the audio content
Alternate Flows	An audio content can be provided in different sample rates, higher number of samples generates higher quality audio content, but also larger files, the versions can be used according to the user preference, or connection rate, or processing capabilities of the device

Use Case Name	**Change Speed**
Use Case Description	The audio content can be played in a higher or lower speed, according to the context of use
Primary Actor	User / System
Precondition	There is audio content, an aimed speed (different of the original one) and it is possible to change it
Trigger	The user preferences, profile, cognitive abilities or context of use (for instance if the user is accessing a content in a language other than her mother tongue, she may prefer to access it in a lower speed)
Basic Flow	Given an audio content, the player has its speed parameter set to a different value
Alternate Flows	There are different documents containing audio files that can be selected and played in different speeds

Use Case Name	**Change Volume**
Use Case Description	The volume of the sound is modified (to a higher or lower level)
Primary Actor	User / System

Precondition	There is audio content, and its volume can be modified
Trigger	The level of noise in the environment may trigger this technique, the user preference, a user hearing impairment
Basic Flow	The value of a parameter in the player of the audio content is modified accordingly
Alternate Flows	There are different audio contents that can be selected, chosen, according to the volume level

Use Case Name	**Convert Channel**
Use Case Description	The audio can be mono channel (left, right), monophonic or stereophonic, and due to the context of the user, it may change
Primary Actor	User / System
Precondition	There is audio and the possibility to convert the original channel type to another one
Trigger	The user preferences, or the platform (sound system available)
Basic Flow	There are some versions of the audio content available with different channel types, an event triggers the change, and the correct type of the audio content is selected to be played
Alternate Flows	

Use Case Name	**Convert Format**
Use Case Description	There is an audio content in a given format, and it can be converted into a different format
Primary Actor	User / System
Precondition	There is an audio content in a given format, and a target format for the conversion, there are versions available for the target format, or methods or processes implemented and available to perform the conversion
Trigger	The device of the user is unable to play the audio content in a given format, the connection rate is not good enough to receive and play the content in its original format, the user has not the required player or codec
Basic Flow	There are different versions of the audio content in different formats and the right one is selected to be played
Alternate Flows	There are methods implemented and available able to perform the conversion

Use Case Name	**Translate Modality**
Use Case Description	Given an audio content it can be converted from audio to text or image for instance
Primary Actor	User / System
Precondition	There is audio content, and alternative versions to provide it in different modalities
Trigger	The user has a hearing impairment that requires alternative versions; the sound system is unavailable or broken; the level of noise in the environment is too high; the user sets its preferences to text (e.g. for privacy concerns, once there is no headset available and she wants to avoid information disclosure)
Basic Flow	There is a version of the content available in an alternative modality

Alternate Flows	There is a method implemented and available to convert the audio content into a different modality

Use Case Name	**Translate Language**
Use Case Description	Given an audio content in a given language, it is translated to another one
Primary Actor	User / System
Precondition	There is audio, and there is a version of the audio content available in another language, or a method implemented that allows the translation
Trigger	The user preferences or the user profile; the location of the user/device; the context of use
Basic Flow	There are versions of the content available in different language
Alternate Flows	There is a method available to perform the translation (e.g., a web service)

Use Case Name	**Simplify**
Use Case Description	The content of the audio is simplified according to certain rules (pre-defined) and a given context of use
Primary Actor	User / System
Precondition	There is audio content (speech, lecture) and there is an alternative version (simplified) of the content, matching with the user context, that can be presented
Trigger	The user has a certain context in which it is not appropriate to present the original version of the audio document (for instance when the content is in a technical level, and the user is not an expert in the topic)
Basic Flow	The context of the user is gathered (for instance he is under a certain age, or educational level), he matches the pre-defined criteria, the audio document is simplified (or there is a version of it stored that can be accessed and re-used), the simplified version of the audio content is presented
Alternate Flows	

Use Case Name	**Summarize**
Use Case Description	Modify the original content to provide a shorter version of it (such as an abstract)
Primary Actor	User / System
Precondition	There is content that is possible to be summarized
Trigger	This technique can be applied when the battery is at a low level, when the connection is weak, when the user has a small screen, when the user has not much time available
Basic Flow	The audio content is selected, and the content is analyzed and is re-defined to produce a summary of the original content
Alternate Flows	

Use Case Name	**Truncate (aka Elision)**
Use Case Description	The content is edited, ‘cut’ or ‘cropped’ at a certain point, in a way that the user only accesses a subsection of it
Primary Actor	User / System

Precondition	There is content and a logic in the system defining what will be removed and what will remain from the original content
Trigger	This technique can be applied when the battery is at a low level, when the connection is weak, when the user needs to access part of the content first (and then be able to access the rest, paying for it for instance)
Basic Flow	The audio content has part of it removed, after a given moment for instance
Alternate Flows	There are short versions available that can be selected to play

Annex C – Use Cases Description for Images

Use Case Name	**Adjust Shape (c.r. Morph)**
Use Case Description	The format of the image is adapted according to the context
Primary Actor	User / System
Precondition	There is an image as context information and the image to be adapted, there is also a function that adjusts the border of the image according to the shape of the image of the context
Trigger	Once the image of the context has been captured, the image of the system can be adapted
Basic Flow	The image of the context is captured (e.g. a picture is taken), the borders are analysed and defined, and the image of the system is adapted accordingly
Alternate Flows	There are samples of images to be chosen as context (e.g. user preferences) and a set of images in the system to be selected according to the shape (instead of adjusting it, the most appropriate is selected)

Use Case Name	**Change Brightness**
Use Case Description	The brightness or luminance of the image is modified according to the context
Primary Actor	User / System
Precondition	There are images composing the UI, there is a function to adjust the brightness level
Trigger	The level of light of the environment, the user preferences
Basic Flow	Given a pre-condition, such as, if the light level in the environment is high, then the level of brightness of the UI is set to another value
Alternate Flows	

Use Case Name	**Change the Color Balance**
Use Case Description	This techniques changes the color balance of a given image in the UI
Primary Actor	User / System
Precondition	There is one or more images in the UI, and a method that allows the color balance to be modified
Trigger	The quality of the image, the type of light in the environment, a visual impairment of the user
Basic Flow	The image is selected, and a technique is applied to change its color balance
Alternate Flows	There are alternative versions of the image available, with different color balances, that can be exhibited

Use Case Name	**Change the Color Map**
Use Case Description	The colors of the images in the UI are modified according to the context
Primary Actor	User / System
Precondition	There is one or more images available at the UI
Trigger	The user preferences, the quality of the image, the type of the screen, the user has a visual impairment, analysis of the image is more easily done using different color mapping (e.g., false colour medical images, astrophotography)

Basic Flow	The images available in the UI are selected, and a new color map is defined, the image is processed and as a result a new version of it is produced with the new color map
Alternate Flows	There are alternative images available containing different color maps, that can be selected and loaded accordingly

Use Case Name	**Collapse to Zoom**
Use Case Description	The collapse-to-zoom is a technique that provides an alternative exploration strategy to allow users to zoom into relevant areas and to collapse areas deemed irrelevant, such as columns containing menus, archive material, or advertising
Primary Actor	User / System
Precondition	There is a set of images, and a method that collapses the images that are less relevant for the end user in a given moment
Trigger	The context of use, the dimensions of the screen, a low connection rate, a gesture of the user (to collapse the content)
Basic Flow	The images are mapped, the logic to collapse them is defined, and less relevant images are collapsed and replaced with a 'link' that allows access to them
Alternate Flows	

Use Case Name	**Color Translation**
Use Case Description	The color translation consists in replacing certain colors in an image by other ones
Primary Actor	User / System
Precondition	There is one or more images composing the UI, and a method that allows the replacement of the colors by other ones (or the replacement of the image itself)
Trigger	The screen device, the user preferences, the context of use, a visual impairment of the user
Basic Flow	The image is selected and processed to have certain colors replaces by other ones
Alternate Flows	There are alternative versions of the image content, that are already processed, that can be loaded instead of the original image

Use Case Name	**Change Contrast**
Use Case Description	The contrast of the image is modified according to the context
Primary Actor	User / System
Precondition	There is one or more images available in the UI, it is possible to change their contrast levels
Trigger	The level of light in the environment, the user preferences, the quality of the images
Basic Flow	The image is selected, the level of contrast is increased or decreased according to the context
Alternate Flows	There are alternative images available, so instead of processing them to change the contrast, an alternative image is loaded according to the level of contrast aimed

Use Case Name	Change Transparency
Use Case Description	The transparency level of an image is modified according to the context
Primary Actor	User / System
Precondition	There is one or more images available in the UI, it is possible to change their transparency levels
Trigger	The user preferences, the context of use, the application domain
Basic Flow	The image is selected, the level of transparency is increased or decreased according to the context of use
Alternate Flows	There are alternative images available, so instead of processing them to change the transparency level, an alternative appropriate image is loaded

Use Case Name	Quantization
Use Case Description	Color quantization reduces the number of colors used in an image, which allows a compression of the file
Primary Actor	User / System
Precondition	There is one or more images available and it is possible to transform its quantization
Trigger	If the user has a low connection rate, and the image to be loaded are too heavy, user preferences, device type
Basic Flow	The image is selected, and processed to have its quantization changed
Alternate Flows	There are alternative versions of the image available that can be selected to be loaded

Use Case Name	Digital Compositing
Use Case Description	A set of images are arranged together to produce a composition
Primary Actor	User / System
Precondition	There is a set of images, and a common context/goal to use them together
Trigger	The context of use, user preferences, application domain
Basic Flow	The set of images is loaded, the system has a logic to re-locate them (or the user does it manually), a final image containing the set is composed and generated as a final result
Alternate Flows	There are already some compositions available that can be presented to the end user, there are some algorithms that can be used to rearrange the images, the user selects a given shape to guide the location of the images, the images can be overlapped or not

Use Case Name	Matte
Use Case Description	It merges two or more pictures to produce some final effect
Primary Actor	User / System
Precondition	There is a set of images, and a given criteria to produce a combination of them
Trigger	The context of use, application domain or user preferences
Basic Flow	The set of images is selected, and they are composed to produce some specific effect
Alternate Flows	There is a set of combinations with the images available to be selected and

	presented to the end user

Use Case Name	Optical Compositing
Use Case Description	It consists in composing an image with some special effect, based in a set of other images
Primary Actor	User / System
Precondition	There is a set of images, they can be composed, there is a goal to perform this composition
Trigger	The context of use, application domain or user preference
Basic Flow	The images are loaded, processed and a final image is generated
Alternate Flows	There is a set of images already pre-composed optically that can be loaded and presented for the end user

Use Case Name	Daltonize
Use Case Description	The Daltonize is a technique that transforms the colors of the images in order to allow color blind users to see it with more details.
Primary Actor	User / System
Precondition	There is colored image(s) and the user is color blind
Trigger	Color blind users (namely: protanoia, deuteranopia, and tritanopia), and screens with limited range of colors (e.g. black and white)
Basic Flow	Given an application with a set of images, they are transformed to have the colors accessible for color blind users.
Alternate Flows	There is a set of pre-daltonized images, that can be loaded in case of blind user

Use Case Name	Differentiate
Use Case Description	Image differencing is an adaptation technique used to determine changes between images. The difference between two images is calculated by finding the difference between each pixel in each image, and generating an image based on the result.
Primary Actor	User / System
Precondition	There are two or more images to be compared
Trigger	User preferences, time
Basic Flow	For this technique to work, the two images must first be aligned so that corresponding points coincide, and their photometric values must be made compatible, either by careful calibration, or by post-processing (using color mapping). The complexity of the pre-processing needed before differencing varies with the type of image.
Alternate Flows	The Hutchinson metric can be used to "measure of the discrepancy between two images”.

Use Case Name	Enlarge (aka Re-size)
Use Case Description	An image can be re-scaled in to a larger dimension
Primary Actor	User / System
Precondition	There is one or more images available and space in the UI to be filled

Trigger	The dimension of the screen is larger, the user has a visual impairment, the distance between the user and the screen is far
Basic Flow	In Euclidean geometry, uniform scaling is a linear transformation that enlarges (increases) or shrinks (diminishes) objects by a scale factor that is the same in all directions to preserve aspect ratio. The result of uniform scaling is similar (in the geometric sense) to the original.
Alternate Flows	

Use Case Name	**Interpolate**
Use Case Description	It consists in estimating the value for an intermediate value of the variable. This is often done in when doing image enlargement.
Primary Actor	User / System
Precondition	There is at least one image
Trigger	The need, wish or requirement for a better image in terms of definition and quality, such after a upwards rescaling if the image
Basic Flow	Construct new data points within the range of a discrete set of known data points
Alternate Flows	Sampling, experimentation

Use Case Name	**Geometric Hash**
Use Case Description	It consists in finding 2d shapes in images
Primary Actor	User / System
Precondition	Given two images
Trigger	According to the user preferences, context of use, or application domain
Basic Flow	In an off-line step, the objects are encoded by treating each pairs of points as a geometric basis. The remaining points can be represented in an invariant fashion with respect to this basis using two parameters. For each point, it is quantized; transformed coordinates are stored in the hash table as a key, and indices of the basis points as a value. Then a new pair of basis points is selected, and the process is repeated. In the on-line (recognition) step, randomly selected pairs of data points are considered as candidate bases. For each candidate basis, the remaining data points are encoded according to the basis and possible correspondences from the object are found in the previously constructed table. The candidate basis is accepted if a sufficiently large number of the data points index a consistent object basis.
Alternate Flows	

Use Case Name	**High Dynamic Range Imaging**
Use Case Description	HDR is an imaging technique that is used when the luminance spread is wider than the presentation capabilities of the rendering display (e.g., photography with zones in very deep shadow and others in direct sunlight)’
Primary Actor	User / System
Precondition	
Trigger	Images whose luminance distribution is beyond the dynamic range of the rendering element
Basic Flow	HDR scenes that undergo a HDR process have their luminance values adjusted so they fit within the rendering element limits, in order to enhance the perception or aesthetic values of the scene. Tone Mapping and other

	algorithms may be used to perform the filtering.
Alternate Flows	

Use Case Name	**Morph (c.r. Adjust Shape)**
Use Case Description	Morphing is a special effect in motion pictures and animations that changes (or morphs) one image into another through a seamless transition. Most often it is used to depict one person turning into another through technological means or as part of a fantasy or surreal sequence. Traditionally such a depiction would be achieved through cross-fading techniques on film.
Primary Actor	User / System
Precondition	There is a set of images available, there is a morphing methods implemented
Trigger	The context of use, user preferences
Basic Flow	It involves distorting one image at the same time that it faded into another through marking corresponding points and vectors on the "before" and "after" images used in the morph. For example, one would morph one face into another by marking key points on the first face, such as the contour of the nose or location of an eye, and mark where these same points existed on the second face. The computer would then distort the first face to have the shape of the second face at the same time that it faded the two faces. More sophisticated cross-fading techniques can be employed e.g., vignetting different parts of one image to the other gradually instead of transitioning the entire image at once.
Alternate Flows	

Use Case Name	**Recognize (c.r. Segment)**
Use Case Description	Consists in locating a given object, feature or activity in an image
Primary Actor	User / System
Precondition	There is an image and a target to be searched
Trigger	The context of use, accomplishment of a task
Basic Flow	The image is analysed, the object is searched, the image is scanned and the object of interest is detected and highlighted
Alternate Flows	If the object is not found, the end user is informed (she can look for it in another image, or look for another information)

Use Case Name	**Reduce (c.r. Re-size)**
Use Case Description	Image reduction or elision: reduces or suppresses images from the page, replaces images with a reduced image or the ALT text pointing to the original image.
Primary Actor	User / System
Precondition	There is one or more image, the user has no visual impairment (or assistive technology); images which contain text or numbers can only be reduced by a small amount before their contents become illegible
Trigger	According to the user preferences, visual impairments, connection rate
Basic Flow	Given a set of images they have the size reduced or they are replaced by their alternative text
Alternate Flows	

Use Case Name	**Register**
Use Case Description	Image registration is the process of transforming different sets of data into one coordinate system. Data may be multiple photographs, data from different sensors, from different times, or from different viewpoints. It is used in computer vision, medical imaging, military automatic target recognition, and compiling and analyzing images and data from satellites. Registration is necessary in order to be able to compare or integrate the data obtained from these different measurements.
Primary Actor	User / System
Precondition	There is a set of images from different sources that can be combined
Trigger	Once the images were captured, and according to the user preferences, application domain and context of use
Basic Flow	Transformation models: Image registration algorithms can be classified according to the transformation models they use to relate the target image space to the reference image space. The first broad category of transformation models includes linear transformations, which include translation, rotation, scaling, and other affine transforms. Linear transformations are global in nature, thus, they cannot model local geometric differences between images. The second category of transformations allows 'elastic' or 'nonrigid' transformations. These transformations are capable of locally warping the target image to align with the reference image. Nonrigid transformations include radial basis functions (thin-plate or surface splines, multiquadrics, and compactly-supported transformations), physical continuum models (viscous fluids), and large deformation models (diffeomorphisms).
Alternate Flows	Intensity-based vs. feature-based; Spatial vs. frequency domain methods; Single- vs. multi-modality methods; Automatic vs. interactive methods; Similarity measures for image registration;

Use Case Name	**Rotate (aka Change Orientation, Change Direction)**
Use Case Description	It consists in changing the orientation of the image
Primary Actor	User / System
Precondition	There is an image and a method able to turn it (or alternative images with different orientation schemes)
Trigger	The context of use, application domain, the position of the screen
Basic Flow	The image is selected, its matrix of pixels is rotated in a given number of degrees, the final image is presented to the end user
Alternate Flows	There are alternative images with different orientations that can be presented

Use Case Name	**Segment**
Use Case Description	Segmentation refers to the process of partitioning a digital image into multiple segments (sets of pixels, also known as superpixels). The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze. Image segmentation is typically used to locate objects and boundaries (lines, curves, etc.) in images. More precisely, image segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain visual characteristics.
Primary Actor	User / System
Precondition	There is an image and a given criteria to segment it (such as color or shapes)
Trigger	The context of use, application domain, user preferences

Basic Flow	The image is analysed, processed and different layers of information are produced accordingly
Alternate Flows	

Use Case Name	**SmartView**
Use Case Description	The smart view is a functionality that is integrated into a document viewer. This tool provides the decomposition of a HTML document content into logical sections that can further be selected by the user and viewed independently from the rest of the document. Moreover the layout of the selected portion is modified for optimal reading or for meeting the users or devices requirements. Smart view attempts to solve both the scaling down and the scaling up problems when viewing HTML documents.
Primary Actor	User / System
Precondition	There is a set of contents which access may be optimized
Trigger	The context of use, small screen device
Basic Flow	The contents are analysed and rearranged properly
Alternate Flows	

Use Case Name	**WebThumb**
Use Case Description	The Web Thumb technique for Web Exploration using a Body of Techniques for Handheld Ubiquitous Mobile Browsing, improves the web browsing experience. The approach consists to keep the original design of a website in a thumbnail and to permit to the end-user to put some parts of the website that him interests in other thumbnail.
Primary Actor	User / System
Precondition	There is a set of contents that can be minimized
Trigger	The context of use, a small screen device or a low connection rate
Basic Flow	The content is analysed, detected and the user selects pieces of content to be accessed with thumbs
Alternate Flows	

Annex D – Use Cases Description for Text

Use Case Name	**Add Explanation**
Use Case Description	There is a text content, and additional explanations are prepared and presented to the end user to make the concept more clear for her
Primary Actor	User / System
Precondition	There is a text content, there is alternative explanation available for that concept
Trigger	The context of use, user preferences, application domain
Basic Flow	There is a certain content, alternative explanations are prepared and can be accessed by the end user to comprehend better the concepts of the content
Alternate Flows	A set of links with further information is presented to the end user

Use Case Name	**Align**
Use Case Description	There is a text content, and the text is aligned according to a certain criteria
Primary Actor	User / System
Precondition	There is a text content and it is possible to align it differently
Trigger	The context of use, the application domain, the user preferences
Basic Flow	The content is selected, and a transformation is applied to change its original alignment to a different one
Alternate Flows	There are versions of the same content, with different alignments that can be selected and presented to the end user

Use Case Name	**Altering Fragments**
Use Case Description	There are alternative versions of the same text content that can be presented to the end user according to her actual context
Primary Actor	User / System
Precondition	There is a text content, and alternative versions of it that better suits for certain groups of user
Trigger	The context of use, the user preferences, profiles, educational level, type of device, connection rate, screen dimensions
Basic Flow	The content is defined, the user profile is define, there are alternative versions of the same content that better suits for the actual context of use, the best version is selected and presented to the end user
Alternate Flows	

Use Case Name	**Auto Correct**
Use Case Description	The mistakes of the user while typing are identified and corrected by the systems
Primary Actor	User / System
Precondition	There is a text content, it is possible to verify and correct (or suggest corrections) for it
Trigger	The context of use (e.g. learning a language, writing a letter), application domain (e-learning), the user preferences, the educational level of the user,

	the domain in the language
Basic Flow	The content is selected, a verification is performed, the errors are selected, and a correction is suggested to the end user
Alternate Flows	The correction is performed and the user has to accept or reject it (and evaluate it eventually)

Use Case Name	**Change Background**
Use Case Description	There is a text content, and its background is modified
Primary Actor	User / System
Precondition	There is a given text content, with a background, and it is possible to modify the background
Trigger	The user preferences, context of use, application domain, light level of the environment, or visual impairment of the user
Basic Flow	The background is selected, a new color, or image or document is defined to fill it, the new version is presented to the user
Alternate Flows	There are some options available for the end user to select the color, image or type of texture of the background

Use Case Name	**Change Contrast**
Use Case Description	There is a text content, the colors of the background and the text are modified to produce a different contrast
Primary Actor	User / System
Precondition	There is a text content, it is possible to change the colors of the content and the background
Trigger	The user preferences, the light level in the environment, the user has a visual impairment
Basic Flow	The text is selected and its color is modified, the background color is modified
Alternate Flows	There are alternative versions of the same text content, but with different levels of contrast, and the most appropriate one in a given context is selected to be presented to the end user

Use Case Name	**Change Font Family**
Use Case Description	The text content has its font family modified
Primary Actor	User / System
Precondition	There is a text content, and it is possible to modify its font family (by applying a transformation, or loading an alternative version of the content)
Trigger	The user preferences, the device constraints (e.g. sometimes the original family type of the content is not available to be exhibited to the end user), the application domain, context of use
Basic Flow	The content is selected, the font family is modified, the new version of the content is presented to the end user
Alternate Flows	There are alternative versions of the same content with different font families, and the most appropriate for a given context is selected and presented to the end user

Use Case Name	**Change Font Style**

Use Case Description	There is a text content, in a given font style, the font style is modified by another one
Primary Actor	User / System
Precondition	There is text content, and it is possible to modify it
Trigger	The user preferences, context of use, application domain
Basic Flow	The content is selected, and the font style is modified
Alternate Flows	There are alternative versions of the text, with different font styles, the most appropriate one is selected to be presented

Use Case Name	**Change Orientation**
Use Case Description	The direction of a certain text content is modified (e.g. from portrait to landscape)
Primary Actor	User / System
Precondition	There is a text content, and it is possible to change the orientation of its presentation
Trigger	The orientation of the device, the user preferences, the context of use
Basic Flow	The content is selected, and its orientation scheme is modified
Alternate Flows	There are alternative presentations of the same content with different orientation schemes

Use Case Name	**Change Readability**
Use Case Description	Given a text content, the readability of it is improved by applying different adaptation techniques, such as: improving the contrast, changing the spacing, the font type, the serif, etc.
Primary Actor	User / System
Precondition	There is a text content, and a set of techniques to be applied to make its readability better
Trigger	The context of use, user preferences, type of device, visual impairments, application domain
Basic Flow	The text content is selected, the set of techniques is defined, the new version of the content is generated and presented to the end user
Alternate Flows	There are alternative versions of the content that have different readability levels that can be selected and presented according to the context of the user

Use Case Name	**Change Serif**
Use Case Description	There is a text content, the type of serif is modified (present / absent)
Primary Actor	User / System
Precondition	There is a text content, and a method that modifies the type of font in use to remove or add serif
Trigger	The context of use (e.g. Wikipedia says that low screen resolutions make serif more difficult to discern, decreasing the readability of the text)
Basic Flow	The text content is selected and the type of font is modified to change the serif of it
Alternate Flows	There are alternative versions of the same text content, with different types of serif, that can be selected and presented to the end user

Use Case Name	**Change Size**
Use Case Description	Given a text content, the size of the font is modified (increased or decreased)
Primary Actor	User / System
Precondition	There is a text content, and a given criteria that triggers the adaptation technique
Trigger	The user has a visual impairment, the position of the screen is far from the user, the device has a small or large screen, user preferences
Basic Flow	The content is defined, the size of the font is modified
Alternate Flows	There are alternative versions of the text, with different layouts of the UI, and the most appropriate one, in a given context, can be selected and presented to the end user

Use Case Name	**Change Spacing**
Use Case Description	Given a text content, the spacing between the characters and/or the words are modified
Primary Actor	User / System
Precondition	There is a text content, and a given criteria that modifies the spacing of it
Trigger	The user has a visual impairment, the device has a small or large screen, user preferences
Basic Flow	The content is defined, the amount of spacing is modified
Alternate Flows	There are alternative versions of the text, with different spacing levels, and the most appropriate one, in a given context, can be selected and presented to the end user

Use Case Name	**Compare Explanation**
Use Case Description	Given two explanations about a certain topic, the differences between them are identified and remarked to the end user in a way that she is aware of the differences
Primary Actor	User / System
Precondition	There are two explanations (in text content) about a concept, there is a methods implemented and available to compare and highlight the differences between the two contents
Trigger	The context of use, application domain (e.g. e-learning, news), user preferences, task, period (e.g. each 24 hours)
Basic Flow	There are two given contents, the comparison between them is performed, the similarities and differences are remarked and presented to the end user
Alternate Flows	The differences or similarities can be presented with an explanation (e.g. with numeric information, statistical data)

Use Case Name	**Compare**
Use Case Description	Given two different text contents, they are compared and the differences between them are remarked to the end user
Primary Actor	User / System
Precondition	There are two text contents, and a method to compare and present their differences to the user
Trigger	The user preferences, context of use, refreshments of the content, time

Basic Flow	There are two text contents, they are compared, the differences are searched, found, and remarked
Alternate Flows	Some statistical information can be presented (e.g. number of different words)

Use Case Name	**Describe**
Use Case Description	Given a certain text content (e.g. a word), the description of the content is generated and presented to the end user
Primary Actor	User / System
Precondition	There is a text content, there is a description about one or more concepts of the text
Trigger	The user profile, background, interests, educational level, impairments, the context of use, history of navigation
Basic Flow	There is a certain concept in a text content, a description about it is generated, and presented to the user
Alternate Flows	

Use Case Name	**Dim Fragments**
Use Case Description	The text content has certain parts that are not so relevant in a given context, so they can be dimmed (or have its contrast reduced)
Primary Actor	User / System
Precondition	There is a text content, and a given criteria that defines certain parts of the text to be dimmed
Trigger	The user preference, profile, background
Basic Flow	There is a text content, the criteria is defined, certain parts of the text are dimmed
Alternate Flows	There are alternative versions of the same text content in which certain fragments are dimmed

Use Case Name	**Explain**
Use Case Description	There is a text content and the explanation about the concepts there presented are prepared and exhibited to the end user
Primary Actor	User / System
Precondition	There is a text content, there is explanation about it
Trigger	When the user access the text, user preferences, profile, educational level or background
Basic Flow	There is a text content, and an explanation about it is prepared and exhibited to the user (e.g. by means of a link)
Alternate Flows	

Use Case Name	**Highlight**
Use Case Description	There is a text content, and given a certain criteria, it is completely or partially highlighted
Primary Actor	User / System
Precondition	There is a text content, and a certain criteria to highlight it (e.g. a given

	word)
Trigger	The context of the user (e.g. accessing search results)
Basic Flow	There is the text content, the information of interest is highlighted and presented to the user
Alternate Flows	There is an alternative version of the text content, which has certain words already highlighted, that is selected to load and be presented to the end user

Use Case Name	**Outlining**
Use Case Description	It exhibits to the end user the headers as hyperlinks for the original content
Primary Actor	User / System
Precondition	There is a text content that is organized with headers
Trigger	The end user load a content in a small screen device (such as a PDA, pager or a mobile phone) or the connection rate is low
Basic Flow	Given a content, the structure is analysed and links are provided to the original contents of each section of the document
Alternate Flows	

Use Case Name	**Present Pre-requisite**
Use Case Description	It exhibits to the end user o text content that explains some concepts of the current one, in a way that the pre-requested knowledge of the end user is fulfilled
Primary Actor	User / System
Precondition	There is a text content, there are pre-requisite concepts, there are content to explain the pre-requisites
Trigger	The accomplishment of a task by the user, the user preferences, educational level, background information, history of navigation
Basic Flow	Given a content, the pre-requisite information about it, is prepared and made available for the end user
Alternate Flows	

Use Case Name	**Find Similarity**
Use Case Description	Given two different texts, this techniques look for similarities between the contents and highlight it to the end user
Primary Actor	User / System
Precondition	There are two (or more) given texts, there is a method implemented to find the similarity between them, and highlight it to the end user
Trigger	The user preference, difference in time (e.g. each 24 hours)
Basic Flow	Given two text contents, the similarities between them are searched and highlighted to the end user (for instance remarking the color of the text)
Alternate Flows	The differences between two contents can be remarked

Use Case Name	**Simplify**
Use Case Description	The content is presented to the user in a simplified version, according to a given pre-defined criteria (e.g. educational level of the end user, or impairments)

Primary Actor	User / System
Precondition	There is text content, it can be simplified, there is a method to simplify it, a criteria to guide this process, or alternative version of the text content with varied versions of simplicity
Trigger	The context of the use (preferences, profile, education level, cognitive level), the connection rate, the dimension of the screen
Basic Flow	The text content is defined and selected, and it is processed to generate a simplified version of it, the simplified version is presented to the user
Alternate Flows	There are versions of the same text in different levels of simplicity

Use Case Name	**Stretch Text**
Use Case Description	Given a text content, some parts of it are selected to be presented to the end user, and the remaining is hidden, and can be accessed by a link
Primary Actor	User / System
Precondition	There is text content and a logic defining which parts can be exhibited or hidden
Trigger	The context of use, the user preferences, the application domain
Basic Flow	The content is selected and analysed, parts of it are marked to be exhibited, parts are hidden and replaced by links to the original content
Alternate Flows	

Use Case Name	**Summarize**
Use Case Description	The text content is summarized to generate an abstract of the content
Primary Actor	User / System
Precondition	There is text content, and a method to process it and to generate abstract versions, according to a given criteria (e.g. the number of words)
Trigger	The context of the user, a low connection rate, a small screen device
Basic Flow	The content is select and processed, and a new version of it is generated to produce an abstract of it, according to a given criteria
Alternate Flows	There are alternative versions of the content available and the most appropriate one, is selected to be presented to the end user, according to a given criteria

Use Case Name	**Sort**
Use Case Description	The text content is ordered according to a given criteria (e.g. alphabetical order)
Primary Actor	User / System
Precondition	There is one or more text contents, that are re-ordered to be presented according to the criteria that is already pre-defined
Trigger	The context of use, application domain, user preferences
Basic Flow	The content is selected, the criteria to order is defined, the content is sorted according to the criteria, and then presented to the user in an alternative ordering
Alternate Flows	There are multiple versions of the content available, in different orders, and the most appropriate one, according to the context of the user, is exhibited in the UI

Use Case Name	**Translate**
Use Case Description	The text content is in a given language, and the goal is to translate it to another one
Primary Actor	User / System
Precondition	There is a text content, and an aimed language for the translation, that is different from the original one
Trigger	The context of the user, the location, user preferences
Basic Flow	The content is selected, translated, and presented in a language that is different from the original one
Alternate Flows	There are versions of the text available in different languages, the appropriate one is selected to be presented (according to the context)

Use Case Name	**Truncate**
Use Case Description	The text content is truncated when only a sample of it is presented in the UI
Primary Actor	User / System
Precondition	There is one or more text contents composing the UI, and a certain criteria defined to truncate it (e.g. a number of words, or characters)
Trigger	The dimension of the screen, the connection rate of the user, the user preferences, the amount of content to be displayed
Basic Flow	The text content is selected, part of it is removed, and the remaining is presented
Alternate Flows	There are samples of the text with different lengths and an appropriate one is selected to be exhibited (according to the context)

Use Case Name	**Zoom**
Use Case Description	This technique changes the level of zoom in a text content, in order to make the font size bigger or smaller according to the context
Primary Actor	User / System
Precondition	There is one or more text contents available and a method implemented that allows the level of zoom to change
Trigger	The use has a visual impairment, the content is presented in a small size, the device (screen) is far from the end user
Basic Flow	The text content is defined, selected and has its zoom level modified
Alternate Flows	

Annex E – Use Cases Description for Video

Use Case Name	**Change Frame Resolution**
Use Case Description	It consists in modifying the rate between pixels and lines. Given that a video frame is composed of lines. In digital video, each line is sampled to create a number of pixels (samples) per line. The more lines per frame, the higher the image resolution. The more pixels per line, the higher the resolution of each line.
Primary Actor	User / System
Precondition	There is a video content, composed by a set of multiple frames; its frame resolution is known and the target one too; there is also a method implemented and available to perform the conversion
Trigger	The screen of the device of the user
Basic Flow	The properties of the current video are gathered The target resolution is read The possibility of the conversion is checked If possible, the frames are all processed to be converted to the aimed frame resolution
Alternate Flows	There are multiple versions of the video content available at the server The right one (according to the new context of use) is selected and loaded

Use Case Name	**Change Spatial Quality**
Use Case Description	The spatial quality refers to the quality of the image regarding definition type of a frame, higher resolution videos have a better quality of image than lower resolution ones
Primary Actor	User / System
Precondition	There is a video content, the original and the target spatial quality is known, it is possible to perform the transformation between them
Trigger	The context of use, user preferences, the device of the user, screen dimension, connection bandwidth
Basic Flow	The video content is available and its frames can be processed by a method to change the original spatial quality, to the aimed one
Alternate Flows	There are different versions of the video content available, with different spatial qualities

Use Case Name	**Skip Frame**
Use Case Description	It consists in removing during the presentation of a video content one or more frames of it, according to a given criteria
Primary Actor	User / System
Precondition	There is a video content, a given criteria to remove frames, and a method to process the video
Trigger	The context of use, processing capabilities, network bandwidth
Basic Flow	The criteria is known, the video content is available The content is processed accordingly with a method, to have certain frames removed

	The new version of the video content is then presented to the user
Alternate Flows	The 'player' can be set to skip automatically certain frames according to the criteria specified Or, there are multiple versions of the video available (with certain frames already processed)

Use Case Name	**Summarize**
Use Case Description	It consists in creating a shorter summarized version (e.g., such as a movie 'trailer') of a given video content
Primary Actor	User / System
Precondition	There is a video content to be exhibited
Trigger	The context of use, user wishes
Basic Flow	The video can be processed to generated automatically a summarized versions of it, i.e. with reduced time, frames
Alternate Flows	There are multiple versions of the video available to be selected and presented accordingly

Use Case Name	**Remove Shot**
Use Case Description	It consists in deleting a specific shot of a video content; given that a shot is a continuous action (in time and space) produced by the perspective of a single camera
Primary Actor	User / System
Precondition	There is a video content, composed by one or multiple shots
Trigger	The context of use, e.g. user wish, requirements
Basic Flow	The content is known, the initial and final times of the shot are known, the shot is removed from the original content
Alternate Flows	The content is known, the initial and final times of the shot are known, the shot is not presented by the video player Or, there are multiple versions of the original content, that can be presented accordingly

Use Case Name	**Transcode**
Use Case Description	Transcode can decode and encode many video formats, e.g. MPEG-1/2, MPEG-4(-part 3) (also DivX and XviD variants), Quicktime / MPEG-PS (DVD) (decode only), MPEG-1-layer-1/2/3 audio, AC3 audio
Primary Actor	User / System
Precondition	The video content is available, the original and the target formats are known, it is possible to perform the transcoding
Trigger	This is usually done in cases where a target device does not support the format or has limited storage capacity that mandates a reduced file size, or to convert incompatible or obsolete data to a better-supported or modern format.
Basic Flow	The video content is known, and is available for processing, the original an the target formats are known, and there is a method implemented and available that performs the conversion process: the most popular transcoding process is a two-step process in which the original content is decoded to an intermediate uncompressed format (i.e. YUV for video), which is then

	encoded into the target format.
Alternate Flows	There are alternative versions of the video content available to be presented

Use Case Name	**Select (c.r. Skip Frame)**
Use Case Description	It consists in selecting video contents, or parts of them from a set of options according to a given criteria, e.g. frames containing a certain animal type are selected to be presented to the end user
Primary Actor	User / System
Precondition	There is a video content and a given criteria to select parts of it
Trigger	The context of the user
Basic Flow	The criteria is identified, the video content is processed and a new version of it is presented to the end user (according to the selection criteria)
Alternate Flows	

Use Case Name	**Replace**
Use Case Description	It consists in replacing a video content by a similar alternative, as a text description
Primary Actor	User / System
Precondition	There is a video content and a similar equivalent option to replace it
Trigger	The context of the use, device type, interaction modality, user impairment, disabilities
Basic Flow	The video content is analysed, an alternative for it is found, the logic behind the replacement is implemented and triggered in a given event
Alternate Flows	Users can collaborate suggesting textual descriptions to replace video contents, e.g. video synopsis

Use Case Name	**Reduce**
Use Case Description	A video can be reduced according to a property, such as dimension and/or time
Primary Actor	User / System
Precondition	There is a video content, that can be reduced in a given criteria (previously defined)
Trigger	The context of use, user preferences, screen dimensions
Basic Flow	The content is known, the given criteria as well, the video is processed, and a new and reduced version of it, is presented to the end user
Alternate Flows	There are alternative versions of the video content previously produced and available, that are reduced in certain properties (e.g. file size) and that can be selected and presented to the end user

Use Case Name	**Synthesize**
Use Case Description	It consists in combining two (or more) different contents according to a given criteria
Primary Actor	User / System
Precondition	There are two or more video contents, a given criteria to synthesize them,

	and a method implemented and available to perform the operation
Trigger	The context of use, application domain
Basic Flow	The contents are known (are available), the criteria is known (and valid, possible to consider), the videos are processed and a new synthesized version of them is produced and presented to the end user
Alternate Flows	

Annex F – Use Cases Description for UI Element

Use Case Name	**Change Size**
Use Case Description	It consists in re-sizing a user interface element
Primary Actor	User / System
Precondition	There is the UI element and enough space to enlarge it, or in case of reducing it is still possible to see it
Trigger	The context of use: a smaller or larger screen device, a new layout of the app
Basic Flow	The UI element is selected and modified until it reaches the aimed size
Alternate Flows	

Use Case Name	**Replace**
Use Case Description	It consists in replacing a UI element for another one with similar functionality
Primary Actor	User / System
Precondition	There is a UI element and one equivalent to replace it
Trigger	The context of use: a new layout of the app, a new interaction modality
Basic Flow	The UI element is defined and replaced by another one equivalent
Alternate Flows	

Adapt UI Element: Form

Use Case Name	**Adapt Form**
Use Case Description	It consists in adapting the form field according to the business logic and validation of the filled content
Primary Actor	User / System
Precondition	There is a form, and some fields are conditioned to the value of other fields
Trigger	The content that the user inserted (e.g. marital status, or age will trigger certain fields to be filled)
Basic Flow	The logic is implemented, the content that the user inserts is validated, and may trigger some specific fields to be filled
Alternate Flows	

Use Case Name	**Adjust Form**
Use Case Description	It consists in replacing certain elements of the form according to the layout of the system
Primary Actor	User / System
Precondition	There is a form, a set of fields, and they can be replaced by equivalent ones
Trigger	According to the device, the interaction modality, the context of use, user impairments
Basic Flow	The form is analysed and certain elements can be replaced by different ones according to the space available in the UI
Alternate Flows	

Adapt UI Element: TextBox

Use Case Name	**Expand TextBox**
Use Case Description	Depending on the length of the content typed by the user, the element is expanded
Primary Actor	User / System
Precondition	There is a UI elements, and it can be expanded until a given point
Trigger	According to the content that the user provided for submission
Basic Flow	The property of the element is set to be flexible, and its lengths increase according to what the user types
Alternate Flows	

Adapt UI Element: Table

Use Case Name	**Split Table**
Use Case Description	It consists in splitting a table in columns
Primary Actor	User / System
Precondition	There is a table, and not enough space to present it entirely to the end user
Trigger	A small screen device, a larger table, a new layout of the UI
Basic Flow	A threshold is set (e.g. number of a column), and the table is split in two or more parts containing the columns
Alternate Flows	The table could be also split regarding its rows

Use Case Name	**Transform Table**
Use Case Description	The content of the table is split and each cell become a new page
Primary Actor	User / System
Precondition	There is a table containing a collection of cells
Trigger	A small screen device (such as a PDA or a mobile phone or a pager)
Basic Flow	The structure of the table is analysed, the content is extracted, pages are generated, and a link between the contents is provided
Alternate Flows	

Adapt UI Element: ToolBar

Use Case Name	**Split Interface**
Use Case Description	It consists in including an adaptive toolbar for the end user
Primary Actor	User / System
Precondition	There is a history of the user navigation being tracked, the most used features will compose the tool bar
Trigger	The context of use, the interaction of the end user
Basic Flow	The navigation is tracked, and after a certain number of accesses in a given feature, it is promoted to compose the toolbar. The toolbar is generated and presented to the end user.
Alternate Flows	

Use Case Name	**Moving Interface**
Use Case Description	It consists in promoting some features to be accessed directly from the toolbar (instead of requiring 2 clicks, a menu, or a popup)
Primary Actor	User / System
Precondition	There is a toolbar, and some space left to add a button for a 'popular' feature
Trigger	The interaction of the end user, a certain number of accesses to a given functionality
Basic Flow	The history of interaction is tracked, after a certain point, a feature is promoted to be presented as a button in the toolbar
Alternate Flows	

Use Case Name	**Visual PopOut Interface**
Use Case Description	The most used features in a toolbar are highlighted for the end user
Primary Actor	User / System
Precondition	There is a toolbar menu, and the history of interaction of the end user presents some tend
Trigger	The interaction steps, user preferences, context of use, application domain
Basic Flow	The history of interaction of the user is tracked and analysed, after a certain number of accesses in a given functionality, its access is highlighted (or remarked)
Alternate Flows	

Annex G – Use Cases Description for Presentation

Use Case Name	**Attach**
Use Case Description	Attach is a property in which part of an interface can be attached to another interface being used so as to recompose another one on-demand. "Any UI component of interest can be attached back to its previously detached UI or to any other UI. Thanks to the attachability property, it is possible to support a UI development process by copy/paste. In traditional visual programming, any UI is drawn by composition of widgets dragged from a tool palette onto a working area. It is possible to copy/paste parts of the widgets, but there is a need to redraw everything. When a UI component is attached to another UI component, they are automatically merged so as to create an entirely new UI.
Primary Actor	User / System
Precondition	There is a logic behind the application that allows users to drag and drop components of the original application to the other, not only the widget needs to be recognized, but also its functionality
Trigger	According to user's needs, task requirements, when a new application is loaded
Basic Flow	Given an interface, a new one (entirely or partly) is merged to it. Any selected component from one UI can be copied, dragged and dropped into another UI to compose a new UI by merging functions, with the sum of functions provided by the individual components.
Alternate Flows	

Use Case Name	**Augment**
Use Case Description	It consists in providing to the user a higher level of detail in the UI
Primary Actor	User / System
Precondition	There is an application, and more details about the interaction and the interface itself are available and can be provided
Trigger	The expertise level of the user, a high error rate during the interaction
Basic Flow	The interface is analysed, more details are created, and in case of difficulty during the interaction the user has it provided
Alternate Flows	The details can be collaboratively created by other users

Use Case Name	**Collapse to Zoom**
Use Case Description	Given an interface, composed of interactors, the original position of the resources is modified, according to pre-defined criteria. In addition to allowing users to zoom into relevant areas, collapse-to-zoom allows users to collapse areas deemed irrelevant, such as columns containing menus, archive material, or advertising. Collapsing content causes all remaining content to expand in size causing it to reveal more detail, which increases the user’s chance of identifying relevant content.
Primary Actor	User / System
Precondition	An interface, composed of interactors, pre-defined criteria to collapse the content
Trigger	The user has a small screen device, there is a lot of content to be presented
Basic Flow	The application recognizes certain interaction of the user, and collapses content that the user considered as less relevant (the content can be later

	retrieved in its original format), the remain of the content is preserved and its layout change to optimize the use of the UI that becomes free
Alternate Flows	

Use Case Name	**Detach**
Use Case Description	Part of an existing interface can be removed from the current interface. This part can be used to recompose another interface on-demand, according to user's needs and task requirements for instance. Detaching is the property of splitting a part of a UI for transferring it onto another platform.
Primary Actor	User / System
Precondition	There is a method that allows users to detach part of the UI and transfer it to another UI
Trigger	According to user's needs and task requirements for instance
Basic Flow	Detachable user interfaces consist of graphical user interfaces whose parts or whole can be detached at run-time from their host, migrated onto another computing platform while carrying out the task, possibly adapted to the new platform and attached to the target platform in a peer-to-peer fashion.
Alternate Flows	

Use Case Name	**Distribute**
Use Case Description	A Distributed User Interface (DUI) consists of a UI having the ability to place some parts or the whole of its components across multiple monitors, devices, platforms, displays, and/ or users
Primary Actor	User / System
Precondition	The application allows part of it to be distributed between devices
Trigger	When a new device is connected, according to the user preferences, task, context of use, application domain
Basic Flow	The application is composed by resources that may be distributed, once there is the possibility to distribute this resources, the context is considered and adaptation rules are applied, the final version of the application contains resources distributed in different domains (such as: platforms, devices or users)
Alternate Flows	

Use Case Name	**Expand**
Use Case Description	It consists in adjusting the size of the UI element according to the user need, so for instance a long content requires a scroll bar to be completely accessed, as well as input elements may have their sizes increased to allow users to see what they are typing
Primary Actor	User / System
Precondition	The elements are able to recognize the content length and to adjust their properties
Trigger	The amount of information (typed, loaded, available…)
Basic Flow	The size of the content is identified and the element is made longer, or the UI has its features adjusted properly (e.g. by making a scroll bar available)
Alternate Flows	

Use Case Name	**Fish Eye (aka Distorted View Approach)**
Use Case Description	It consists in providing a visualization mechanism to the user, that allows her to see a specific part of the UI (usually a circle) with higher level of details than the remaining part
Primary Actor	User / System
Precondition	There is the fish eye mechanism implemented and available for interaction
Trigger	The user has a small screen device, such as a pager, mobile phone, and a lot of content to be accessed
Basic Flow	The content in a certain region is zoomed in and the rest has its size decreased (zoomed out)
Alternate Flows	

Use Case Name	**Full Screen**
Use Case Description	It consists in optimizing the use of the screen, changing the layout to fit completely within the space available
Primary Actor	User / System
Precondition	There is more space available in the screen than the UI is actually occupying
Trigger	The user has a large screen device, a new device is connected, according to the user preferences, context of use, or application domain
Basic Flow	The dimension available is identified, and the resources of the UI are re-located in order to fill it completely
Alternate Flows	

Use Case Name	**Large Screen**
Use Case Description	Given a large screen (wide) the content of the page (and all resources available) must be adapted in order to accommodate better in the interface
Primary Actor	User / System
Precondition	The user has a large screen device
Trigger	The screen dimension, the device of the user, the user preferences, context of use, or application domain
Basic Flow	The UI resources are re-distributed in order to fill completely the dimensions of the screen available
Alternate Flows	The content can be also augmented, or re-sized

Use Case Name	**Migrate**
Use Case Description	Migration consists of transferring any UI component (presentation and dialogue states) from one platform to another.
Primary Actor	User / System
Precondition	There is a method implemented and available that allows users to transfer part of the application to another platform
Trigger	The user has not enough processing capabilities, context of use, application domain
Basic Flow	An specific approach considers migration of applications shared among several users: when migration is triggered the environment starts a fresh

	copy of the application process in the target system, and replays the saved sequence of input events to the copy in order to ensure that the process will get the state where it left off.
Alternate Flows	Another approach assumes that the desktop version of an application exists, without posing any restriction on the method or tool used for its development. Then, during the user session, a version for the platform is dynamically generated exploiting model-based techniques

Use Case Name	**Mirror**
Use Case Description	It consists in changing the presentation of the UI in an inverted layout
Primary Actor	User / System
Precondition	There is a method implemented and available that changes the orientation of the layout
Trigger	The context of the user (culture), reading style (direction)
Basic Flow	The UI elements are detected, inverted and presented to the user with a different layout
Alternate Flows	

Use Case Name	**Optimize**
Use Case Description	Optimize consists in using the space available in the interface in the best possible way.
Primary Actor	User / System
Precondition	The layout of the UI can be re-arranged
Trigger	The user has a larger screen
Basic Flow	Given an interface, and the dimensions of the screen the 'window', the content is processed in order to fill the screen in a more optimized way.
Alternate Flows	

Use Case Name	**Overview + Detail (aka Mini-Map)**
Use Case Description	Overview + detail splits a Web page into multiple sections and provides an overview page with links to these sections
Primary Actor	User / System
Precondition	There is a method implemented to recognize the sections of an application, summarize it, and then present it to the end user linking it to the original content
Trigger	When there is a smaller screen or the user wants to access quickly the content
Basic Flow	Given an application, its content is divided in parts, they are 'named', and only the names are displayed and linked to the original content
Alternate Flows	

Use Case Name	**Print**
Use Case Description	It consists in preparing the UI to be printed
Primary Actor	User / System

Precondition	The layout can be modified to better suit in a printed page
Trigger	The user is going to print the page
Basic Flow	The resources are identified and their layout is modified to better fit in the page to be printed (e.g. to make UI more printable, simplify it by removing extraneous columns and navigation and by combining content split across multiple pages)
Alternate Flows	

Use Case Name	**Re-distribute**
Use Case Description	UI re-distribution, i.e. the application of objects re-distribution to UI components, denotes the re-allocation of the UI components of the interactive space to different interaction resources.
Primary Actor	User / System
Precondition	There is a set of resources composing the UI that can be re-arranged in their layout
Trigger	The context of use, the devices that are available, the loaded applications
Basic Flow	Given an interface, composed of interactors, the original position of the resources is modified, according to pre-defined criteria.
Alternate Flows	

Use Case Name	**Re-mold**
Use Case Description	It consists in reshaping objects without distorting their role. Applied to user interface components, UI re-molding denotes the reconfiguration of the UI that is perceivable to the user and that results from transformations applied to the source UI. Re-molding may result in using different modalities, or in from the suppression and/or exploiting multimodality differently
Primary Actor	User / System
Precondition	There is a set of resources composing the UI
Trigger	The user has a new screen, with a different dimension, according to the context of use, the application domain, or user preferences
Basic Flow	Given an interface, composed of resources, such as interactors, they can be inserted or suppressed, according to pre-defined rules, in order to accommodate better in a new context.
Alternate Flows	UI transformations include: suppression of the UI components that become irrelevant in the new context of use; insertion of new UI components to provide access to new services relevant in the new context of use, reorganization of UI components by revisiting their spatial temporal dependency. Reorganization may result layout and/or their insertion of UI components.

Use Case Name	**Remove Script (aka No-script)**
Use Case Description	It consists in removing the scripts of the application according to the context of use
Primary Actor	User / System
Precondition	The script can be disabled without information lost, the script can be replaced
Trigger	The user has a browser that does not support the script type, the connection rate is too low, the user uses assistive technology (screen reader)

Basic Flow	The script is disabled, and replaced by an equivalent resource
Alternate Flows	

Use Case Name	**Replace**
Use Case Description	It consists in replacing the resource of the UI by another one with equivalent functionality
Primary Actor	User / System
Precondition	There is a set of resources, that can be replaced by equivalent ones
Trigger	The context of use, the type of device: a smaller or a larger screen, the interaction modality
Basic Flow	The resource is defined, an equivalent is selected, and presented instead of the original one
Alternate Flows	

Use Case Name	**Re-size**
Use Case Description	It consists in changing the dimension of the resource, or of the complete UI
Primary Actor	User / System
Precondition	There is a set of resources, that can be re-sized
Trigger	The context of use, a different screen dimension
Basic Flow	The resource is selected and its dimensions are modified
Alternate Flows	The complete UI is re-sized

Use Case Name	**Single Column**
Use Case Description	The content of the application has its layout modified to fit in a single column
Primary Actor	User / System
Precondition	There is content, usually displayed in a large layout
Trigger	The user has a vertical screen, a small screen device
Basic Flow	The content is selected and its layout is modified to fit in a single column
Alternate Flows	

Annex H – Use Cases Description for Navigation

Use Case Name	**Accessible Navigation**
Use Case Description	Applications need to be designed for access and use by everyone, regardless of physical ability or computer capability
Primary Actor	User / System
Precondition	The application has an interaction flow
Trigger	User and platform. People with audio, visual, motor, or cognitive disabilities find it difficult to use Web sites that are not explicitly designed with their accessibility in mind.
Basic Flow	Use a section category layout consistently throughout your site, with the same navigation elements, giving end users a strong sense that they have arrived at a new section and a clear idea of how to get back
Alternate Flows	

Use Case Name	**Annotation of Links**
Use Case Description	The links are augmented with the addition of notes, that remark their relevancy or context
Primary Actor	User / System
Precondition	There is a set of links that the end user can access while interacting
Trigger	The expertise level of the user, the error rate
Basic Flow	The links must be analysed and additional content created and related to them, the content is then presented to the end user
Alternate Flows	

Use Case Name	**Alphabetical Organization**
Use Case Description	It consists in re-ordering the content of the UI in an alphabetical order
Primary Actor	User / System
Precondition	There is a set of contents, with literal description, that can be ordered alphabetically
Trigger	The error rate (when the user is searching for an information and has trouble to find it properly)
Basic Flow	The content are selected, re-ordered and presented to the end user in the UI in the alphabetical order
Alternate Flows	

Use Case Name	**Alternative Navigation**
Use Case Description	It consists in designing an application with multiple, and sometimes redundant, ways of navigating.
Primary Actor	User / System
Precondition	There is a navigation scheme in the application that can be replicated
Trigger	There is free space in the screen, the level of expertise of the user, the context of use, the application domain

Basic Flow	To ensure that end users complete their goals, place search and browse navigation tools at the top and start of the page. Position next-step navigation tools toward the top, but opposite the start, as well as at the bottom. Include navigation tools that relate and promote so that users find things that they might otherwise miss, but position these tools further down the page.
Alternate Flows	Allow users to access resources with shortcuts may optimize navigation for expert users

Use Case Name	**Chronological Navigation**
Use Case Description	There is a set of content in the application that can be ordered chronologically
Primary Actor	User / System
Precondition	The content can be ordered in a chronological way
Trigger	The error rate, the context of use, application domain
Basic Flow	The content is selected, and ordered chronologically be certain property values (e.g. creation date)
Alternate Flows	

Use Case Name	**Concern-Sensitive Navigation**
Use Case Description	An application adapts its content, operations, and links according to the actual situation.
Primary Actor	User / System
Precondition	There is a set of resources whose interaction can be grouped and optimized
Trigger	The application domain and context of use (tasks)
Basic Flow	The functionalities are analysed, and certain resources are grouped to optimize the interaction (improving the efficiency)
Alternate Flows	

Use Case Name	**Categorization**
Use Case Description	Applications need to present sections of information in distinct ways to differentiate them, but these sections must also retain some similarity so that users know they are still on the same site. Category pages keep users oriented.
Primary Actor	User / System
Precondition	There is a set of resources that given a certain criteria can be grouped
Trigger	The context of use, application domain, navigation scheme
Basic Flow	The resources are analysed, grouped and categories are presented to the user
Alternate Flows	The categorization is done collaboratively by end users (e.g., folksonomy)

Use Case Name	**Direct Guidance (c.r. Suggest, Recommend)**
Use Case Description	This technique decides and presents to the user the next best option for interaction, according to the user's goal and other parameters being considered.

Primary Actor	User / System
Precondition	There is a navigation scheme, and a given criteria to present links in a certain order for the users
Trigger	The context of use, application domain, error rate, the level of expertize of the user
Basic Flow	Given a pre-defined criteria, the system suggests to the user the next steps that she should follow (criteria can take into account previous interactions recorded in log file, for instance)
Alternate Flows	

Use Case Name	**Generation of Links**
Use Case Description	The system may discover new useful links between pages and add them, the system may use previous navigation or page similarity to add links, and generating a list of links is typical in information retrieval and filtering systems
Primary Actor	User / System
Precondition	There is already a set of links in the application and the user tend to some interaction sequence
Trigger	An analysis of the previous interactions of the user
Basic Flow	Given two (or more) different contents in a systems, links are generated between them according to defined criteria (such as an analysis of the previous interactions of the user)
Alternate Flows	

Use Case Name	**Hiding of Links (c.r. Dim)**
Use Case Description	This technique restricts the navigation space by hiding links to "not relevant" pages. Pure hiding means the link anchor is shown as normal text (the user cannot see it as a link). Link disabling means that the link does not work; it may or may not still be shown as if there is a link. Link removal means the link anchor is removed (and consequently it cannot be used). A combination of hiding and disabling is also possible. It indicates that the link anchor text is just plain text. Basically, in link hiding the navigation is simplified and restricting the navigational space supports the user's orientation.
Primary Actor	User / System
Precondition	There is a set of links, and a criteria to denote their relevancy
Trigger	The interaction of the user (history)
Basic Flow	Given a set of links in a system, a criteria is established to define which ones are relevant or not, according to the context, links that are considered as not relevant in that context of use are hidden
Alternate Flows	The links can be also dimmed or demoted

Use Case Name	**Hierarchical Organization**
Use Case Description	End users need help when sifting through large amounts of information. Organizing information into a hierarchy makes this task easier.
Primary Actor	User / System

Precondition	There is a set of resources that can be organized hierarchically
Trigger	The error rate, the context of use, the application domain
Basic Flow	The information is analysed and organized hierarchically before presented to the end user
Alternate Flows	Build a hierarchy of categories with input from users or from experts known for good communication skills in the subject area. Use descriptive category names that are distinctive from one another. Use techniques such as card sorting to develop the categories and labels, and use techniques like category identification and category description to test. Repeat items in multiple categories where it makes sense. Keep the maximum number of subcategories per category below 50, and avoid generic terms like miscellaneous.

Use Case Name	**Hierarchical Structure**
Use Case Description	In order to organize the navigation of an application, one approach consists in analyzing the content (and the context of use) and trying to group in logical sets. The dependencies between the sets can be organized hierarchically, as well as the navigation options of the pages.
Primary Actor	User / System
Precondition	The application has a navigation scheme that can be hierarchically organized
Trigger	According to the context of use (content and its abstraction levels).
Basic Flow	The content and tasks are analyzed and grouped logically in a hierarchical way; links between the parts are provided also following the same principle
Alternate Flows	

Use Case Name	**Linear Structure**
Use Case Description	Given an application, transform the sequence of interaction steps into a linear sequence, for instance as a wizard.
Primary Actor	User / System
Precondition	There is a set of resources to be accessed, that can be linearly organized
Trigger	The context of use, the application domain, the level of expertise of the user
Basic Flow	Take the interaction flow, and link one node to the other, in a way that all nodes need to be sequentially accessed
Alternate Flows	

Use Case Name	**Map Adaptation**
Use Case Description	The map adaptation is a form of adaptive navigation support. In order to give users an idea of the whole hyperspace, and some orientation support regarding where the user is in this space, many applications offer some kind of map.
Primary Actor	User / System
Precondition	There is a complex navigation scheme within the application
Trigger	The error rate, the size of the application (complexity), the level of expertise

	of the end user
Basic Flow	Websites often offer a textual sitemap, mostly because this is easy to generate. A graphical map, preferably based on conceptual relationships rather than link relationships, is a better tool for giving insight into the application's structure. Basically, map adaptation concerns adapting graphical representations of the global and local hyperspace link structure using any of the techniques mentioned in this domain or their combination.
Alternate Flows	

Use Case Name	**Mesh Structure**
Use Case Description	Create the structure of navigation as a graph (allowing links between any pair of nodes)
Primary Actor	User / System
Precondition	There is a navigation scheme within the application
Trigger	The error rate, the application domain, the context of use, the level of expertise of the user
Basic Flow	Given the steps or components of an application, structure the links between each pair in a way that there is no strict sequence
Alternate Flows	

Use Case Name	**Popularity-based Organization**
Use Case Description	The items in the application are organized according to their level of popularity
Primary Actor	User / System
Precondition	There is a set of resources that can be organized according to their popularity level
Trigger	The context of use, the application domain
Basic Flow	The items are analysed, their popularity level is identified, and they are organized accordingly
Alternate Flows	

Use Case Name	**Simplify Input Control**
Use Case Description	Mobile devices have limited input capabilities, making it hard to do things like entering text and scrolling. This adaptation use case tries to improve mobile navigation. Mobile devices have limited input capabilities, making Web page navigation a challenge.
Primary Actor	User / System
Precondition	There is a set of interaction resources within the application whose interaction can be simplified
Trigger	The context of use, application domain, the interaction modality
Basic Flow	The input controls are selected, analysed and replaced (by simpler ones); to make mobile navigation easier, minimize the number of links and buttons on your Web pages and make the position convenient, minimize text entry, avoid pick lists and image maps, and limit rich interactions from AJAX technologies to those that can be easily used on small devices
Alternate Flows	

Use Case Name	**Sort Links (c.r. Organize, Structure)**
Use Case Description	This technique sorts all the links according to the user and user-valuable criteria: the closer to the top, the more relevant the link is.
Primary Actor	User / System
Precondition	There is a set of links in the application that can be ordered
Trigger	The context of use, application domain, user preferences, profile
Basic Flow	Given a set of link options for the user to access, they are ordered according to a pre-defined criteria and then presented to the user, criteria can be: most popular, most accessed, alphabetical order
Alternate Flows	

Use Case Name	**Search (a.k.a. Browsable Content)**
Use Case Description	Organizing the information in a clear, consistent, and useful manner can greatly simplify the user's task of finding information. Organize the information so that users can easily find what they need.
Primary Actor	User / System
Precondition	There is a set of resources available within the application
Trigger	The error rate, the level of expertise of the user
Basic Flow	Organize the content in several ways, in categories that make sense to your customers and in the intuitive ways. Build navigation tools and cues that let users know where they are, where they can go, and how to get back. Build each page with its own reading hierarchy so that users can scan it quickly.
Alternate Flows	

Use Case Name	**Suggest Links**
Use Case Description	The system presents to the user a set of links according to the context of use
Primary Actor	User / System
Precondition	There is a set of links and a given criteria relating them
Trigger	The context of use, application domain, the accomplishment of a task (e.g. buying something, reading an article)
Basic Flow	First, the developer define a criteria for the links to be suggested, then the context of the user is gathered and processed, then suggestions of links are exhibited.
Alternate Flows	The users can suggest items collaboratively among themselves