NEXOF-RA

NESSI Open Framework – Reference Architecture

IST- FP7-216446

Deliverable D1.1c Appendix B

Investigation teams results

N. Tsouroulas (TID)
Jose M. Cantera (TID)
José L. Díaz (TID)

Due date of deliverable: 15/06/2010

Actual submission date: 15/06/2010

This work is partially funded by EU under the grant of IST-FP7-216446.

Change History

Version	Date	Status	Author (Partner)	Description
1.0	15/06/2010	draft	TID	Initial Version

EXECUTIVE SUMMARY

This document reports on the contributions coming from the NEXOF-RA open process driven by WP1. To this aim the investigation teams proposed by WP1 that actually contributed were:

- Declarative Authoring Languages for User Interfaces: A specification of such a language, either developed within a project or available as an open standard, to be adopted by NEXOF-RA.
- Context Model to be adopted by NEXOF-RA (User Profile, Delivery Context, Environment (location, time). This IT will in a Context API to be adopted by NEXOF-RA that supports the Context Model, will be platform independent, generic and Extensible.
- Service Front-End Resources Metadata. A formal, declarative metadata specification to be used to describe Front-End resources (gadgets) in mashup platforms.
- Service Front-End APIs. Standard APIs provided by the Front-End layer that facilitate the development and usage of SFRs.

Regarding to the presentation concern, the NESSI projects WP1 has collaborated with are the followings:

- **MyMobileWeb (Eureka-Celtic Project).** MyMobileWeb is the open source reference implementation of the next generation content and application adaptation platform for the Mobile Web.
- **EzWeb (NESSI Strategic Project).** The EzWeb project pursues the development of an enriched enterprise mash-up platform and the development of key technologies to be employed in building the front end layer of new generation SOA architecture.
- **UsiXML (ITEA2 Project).** UsiXML is an existing XML-compliant mark-up language that describes an UI for multiple contexts of use such as character, graphical, auditory or multimodal interfaces. UsiXML has been recently labelled as an ITEA 2 Project (ITEA Call 4, 2008).
- **FAST (FP7 Call 1 Project).** FAST aims at providing an innovative visual programming environment that will facilitate the development of next-generation composite user interfaces.
- **OPEN (Open Pervasive Environments for migratory iNteractive Services) (FP7 Call 1 Project).** The main goal of the OPEN project is to provide a general and open migratory service platform solution based on a sound and innovative scientific approach developed by a multidisciplinary consortium.

The final outcome is a set of reference models that have been considered relevant for the Presentation concern of NEXOF-RA. The analysis has considered the engineering and the end-user-oriented composition viewpoints. From the engineering viewpoint, we have identified the fundamental models for describing multi-target interactive applications that can be able to adapt to its Context of Use. With respect to the user-oriented composition of personalized service front-ends, an initial set of concepts and relationships is presented.

Document Information

IST Project Number	FP7 – 216446	**Acronym**	NEXOF-RA
Full title	NESSI Open Framework – Reference Architecture		
Project URL	http://www.nexof-ra.eu		
EU Project officer	Arian Zwegers		

Deliverable	**Number**	D1.1c Appendix B	**Title**	Advanced User-Service Interactions contributions to Model and Specifications
Work package	**Number**	WP1	**Title**	Advanced User-Service Interactions

Date of delivery	**Contractual**	15/06/2010	**Actual**	15/06/2010
Status	Version 1, dated 15/06/2010		final ☑	
Nature	Report ☐ Demonstrator ☐ Other ☑			
Abstract (for dissemination)				
Keywords	Internet of Services, User Interface, Service Front-End, Model-Based, Composition, Context-Awareness			

Internal reviewers	Reviewer1 (Organization)			
	Reviewer2 (Organization)			
Authors (Partner)	Author1 (Partner1), Author2 Partner2),			
Responsible Author	José M. Cantera (TID)		**Email**	jmcf@tid.es
	Partner		**Phone**	

TABLE OF CONTENTS

1 CONTRIBUTION TO THE REFERENCE MODEL

1.1 Introduction

This chapter proposes a set of reference models that have been considered relevant for the Presentation concern of NEXOF-RA. It is noteworthy to say that there is no a unique reference model for this facet but a set of them. Thus, depending on the viewpoint considered the models might be different.

Our analysis has considered the engineering and the end-user-oriented composition viewpoints. From the engineering viewpoint, we have identified the fundamental models for describing multi-target interactive applications that can be able to adapt to its Context of Use. With respect to the user-oriented composition of personalized service front-ends, an initial set of concepts and relationships is presented.

The **rationale** for the selection of such two viewpoints is on the fact that we believe that context-sensitive front-ends and user-oriented composition and personalization are the two paramount aspects to be developed in the short term roadmap of the Presentation concern of NEXOF. On the other hand, our decision is reinforced by the fact there are mature and good quality contributions coming both from NESSI Strategic Projects and the cooperating projects that have participated in the Investigation Teams and the Open Construction Process (MyMobileWeb, EzWeb, UsiXML, FAST).

1.2 The Engineering Viewpoint: Interactive Application Models

1.2.1 Introduction and Rationale

Model-Based tools have been investigated since the late 1980's. The goal of these tools is to allow a designer to specify the user interface at a level that is independent from the implementation. The specification is usually shared between a set of representations, called models, each model representing a facet of the interface. The number and types of these models is different from one approach to another.

In our view the most important advantages of Model-Based interfaces are:

- They promote a user-centered and UI-centered development process based on high level abstractions, such as task or domain models.
- Models encourage the usage of a declarative approach allowing developers to concentrate on what the application needs to do, rather than how to do it. As a result the time to market is improved.
- Models facilitate the creation of multi target and context-sensitive user interfaces, which it is a difficult and time consuming task [SEF04]
- Models can be checked for consistency and reused across domains

1.2.2 Unified Reference Framework

The **rationale** for selecting the proposed models can be found in the results of the CAMELEON FP5 Project (FP5-IST4-2000-30104) and the *CAMELEON Unified Reference Framework*. CAMELEON [CAL03], describes a framework

that serves as a reference for classifying user interfaces supporting multiple targets, or multiple contexts of use in the field of context-aware computing.

The CAMELEON Framework provides a unified understanding of context-sensitive user interfaces rather than a prescription of various ways or methods of tackling different steps of development. Rather, the framework structures the development life cycle into four levels of abstraction, from the task specification to the running interface (see Figure 1) [FLO06]:

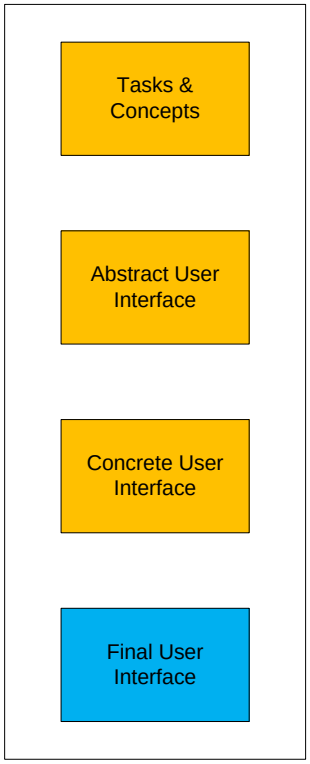

Figure 1 The four abstraction levels

- The *Task and Concepts* level describes the interactive system's specifications in terms of the user tasks to be carried out and the domain objects manipulated by these tasks
- The *Abstract User Interface* (AUI) is an expression of the UI in terms of *interaction spaces* (or presentation units), independently of which interactors are available and even independently of the modality of interaction (graphical, vocal, haptic …). An interaction space is a grouping unit that supports the execution of a set of logically connected tasks.
- The *Concrete User Interface* (CUI) is an expression of the UI in terms of "concrete interactors", which are, in fact, an abstraction of actual UI components (a.k.a widgets) generally included in toolkits. The placement of these concrete interactors is also specified.
- The *Final User Interface* (FUI) consists of source code, in any programming language or mark-up language (e.g. Java or XHTML). It can then be interpreted or compiled. A given piece of code will not always be rendered on the same manner depending on the software environment (virtual machine, browser …). For this reason, we consider

two sublevels of the FUI: the source code and the running interface (see Figure 2)

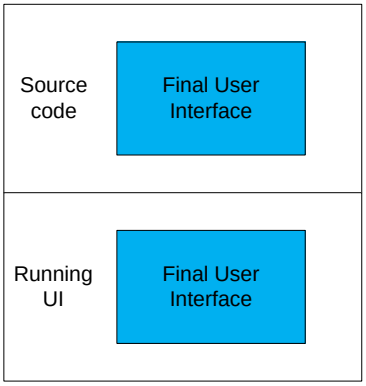

Figure 2 Two sublevels in the FUI

These levels are structured with a relationship of *reification* going from an abstract level to a concrete one and a relationship of *abstraction* going from a concrete level to an abstract one. There can be also a relationship of *translation* between models at the same level of abstraction, but conceived for different contexts of use. These relationships are depicted on Figure 3 Murielle [FLO06]

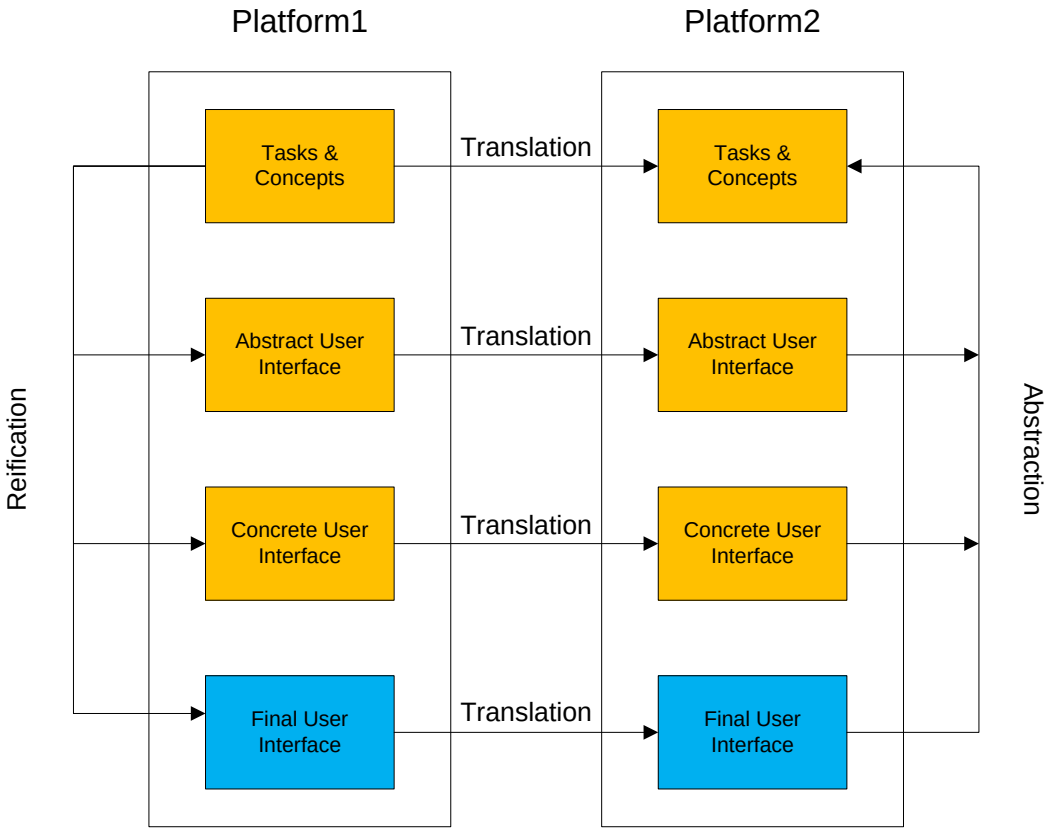

Figure 3 Relationships between components in the Unified Reference Framework

1.2.3 Constituent Models

In accordance with the Unifying Reference Framework, we have identified an aggregation of different models (and its corresponding formal meta-models) that can be used to describe (at a high level of abstraction) interactive applications for NEXOF.

Our proposal is: NEXOF User Interface Specifications will be composed by a combination of models. None of these models would be mandatory and every combination of models would be allowed.

The diagram below illustrates the different constituent models identified and the relationships between them:

First of all, we have identified the main meta-concept which is the Model itself and a few attributes that can be used to its characterization: author, creation date or version. Then, we have distinguished between two different kinds of models:

- Logical Models independent of the description of a user interface:
    - *Context Models* (and meta-models) which describe the main concepts and relationships between contextual entities (see below).
    - *Domain Model*, a description of the information entities (and their relationships) viewed or manipulated by the user. It includes objects storing data that will be displayed or that will be modified by the user, or objects with methods that can be called from the UI or that can modify some aspect of the UI.
- Models that describe the User Interface itself (UiModels):
    - *Task Model*, a formal description of the tasks carried out by a user in interaction with the system. This model can be understood as a generic representation of the envisioned scenarios that were elicited during the requirements analysis. It has generally a hierarchical structure and additional constraints and information about the tasks (such as ordering, launching conditions, associated objects and functions) can be added.
    - *Aui Model* (Abstract User Interface) which describes canonically a user interface in terms of abstract interactors, containers and their relationships, in a way that is independent from the concrete interactors available on the targets. In practical terms this means that an abstract user interface model is platform and modality-independent.
    - *Cui Model* (Concrete User Interface) is a detailed specification of the appearance and behaviour of the UI elements for a concrete platform (desktop, PDA, mobile, etc.) and / or modality (visual, aural …). Although a Concrete UI Model makes explicit the final look and feel of the user interface, it is still a description independent of the toolkit or the execution environment (Java-Swing, Web, VoiceXML …) on which the UI will be finally executed.

- *Behaviour Model* which describes the set of reactions of the user interface to *events,* such as user interactions, changes in the system state, period of time elapsed, and the like. These events trigger *actions*, such as method call or a transition to a target container, provided that certain conditions are met [FLO06].
- *Mapping Model* serves to establish relationships between models or elements of models (for example, between a task belonging to the task model and the widget of the CUI that permits the execution of this task). [FLO06]

There are in the literature different proposals for representing the models described above. WP1 has promoted the creation of different Investigation Teams devoted to study, unify and consolidate (in a collaborative open process) a set of revised and widely accepted models for Interactive Applications. The results of such Investigation Teams are described in the corresponding section.

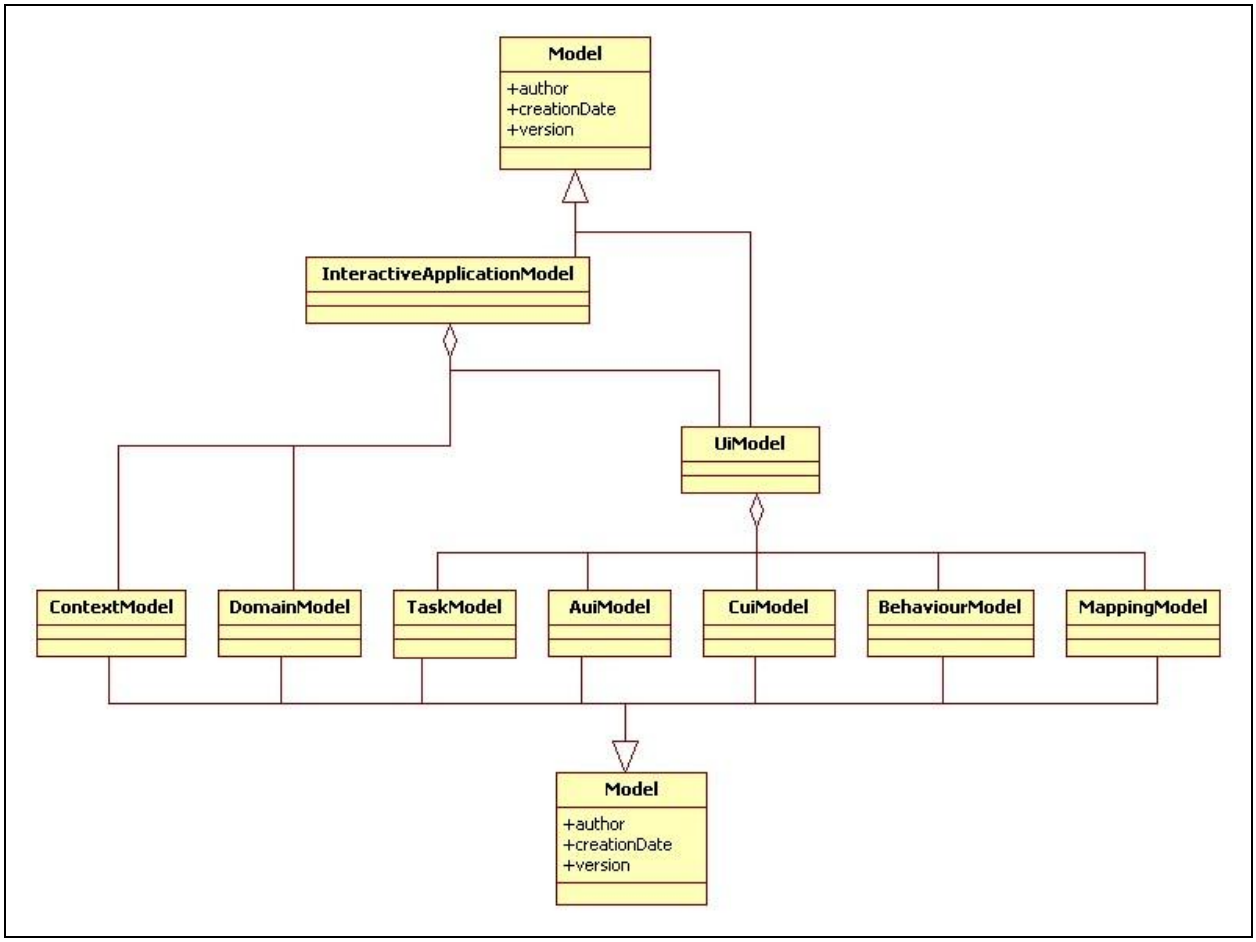

Figure 4 Constituent models for Interactive Applications

1.2.4 Context Models

Context is an all-embracing term. Composed of “con” (with) and “text”, context refers to the meaning that must be inferred from the adjacent text. As a result, to be operational, context can only be defined in relation to a purpose, or finality [CRO02].

WP1 proposes the following levels of Context Description:

- A “**Universal Context Meta-model**” which it is an abstract representation of what is “Context” in the general sense of context-aware computing. As such, it can be reified to more concrete and convenient representations of Context on different domains (user interface, ambient intelligence, Internet of Services, Internet of Things, etc).
- A “**Context Of Use Meta-model**”, which captures the “Context of Use of an interactive system”. This is a Meta-model that it is clearly biased to the Human Computer Interaction (HCI) domain. Such meta-model, in accordance with the Unifying Reference Framework, is itself composed by three different but related sub-models:
  - The “**Delivery Context Model**” a.k.a. (Platform Model) which provides a formal model of the characteristics (and situation) of the hardware, software and network platform(s), that is, the computational, interaction and communication Components that can be used for perceiving and interacting with a NEXOF-compliant system.
  - The “**Environment Model**” which describes any property of interest of the global environment where the interaction takes place. The properties might be physical (e.g. lighting or noise condition) or psychological (e.g. level of stress)
  - The “**User Model**” captures all the variables and characteristics that have to do with the human being (or a human stereotype) which it is interacting with the application. The characteristics modelled or relevant can be very dependant on the application domain. Specific examples are age, level of experience, the permissions, preferences, tastes, disabilities, short term interests, long term interests, etc.

Context Meta-Model

This meta-model is aimed at providing a general purpose representation of Context. In the domain of context-aware computing, Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves. [DEY01].

The figure below depicts a UML representation of this meta-model.

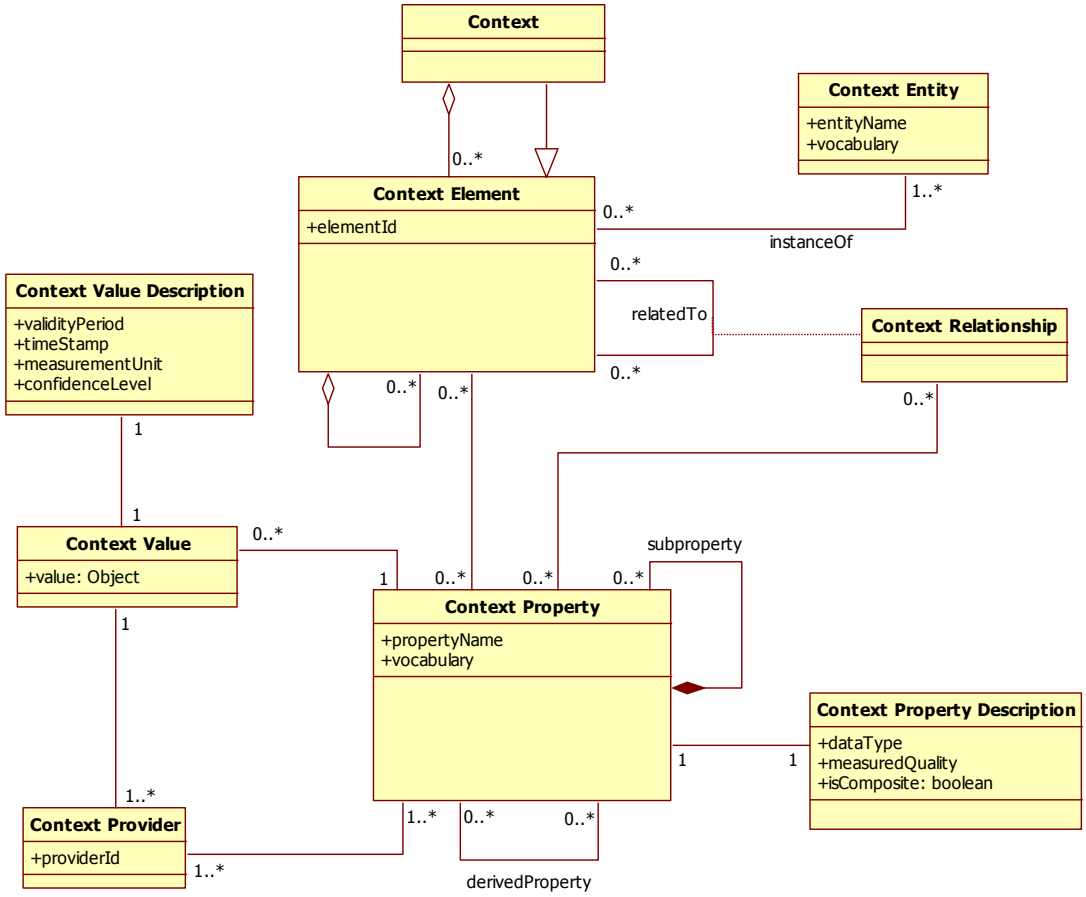

Figure 5 Universal Context Meta-Model diagram

Entity: Context

Description: This class represents the Context itself as a whole composed by the aggregation of different Context Elements (modelled as instances of Context Entities).

Entity: Context Entity

Description: This class represents types of context entities. It can be 'User', 'Object', 'Device' and the like.

Entity: Context Element

Description: A Context Element denotes a specific instance of a Context Entity. Context Elements have different properties, which values are actually the information that characterize their situation.

Attributes:

- elementId: This attribute is the identifier of the element in question.

Entity: Context Relationship

Description: This class denotes a relationship between two Context Elements. Such a relationship is subjected to be associated to certain Context Properties. For example, the relationship between a web browser and a device that can execute such a browser can be modelled as a Context Relationship. Such a relationship can have different properties. For instance, the image formats that are supported when such a combination is present.

Entity: Context Property

Description: This class represents a property of a Context Element, which represents a characteristic of such Context Element or information about its situation. A Context Property may have zero or more values. Examples of properties are: 'position' or 'cpuSpeed'. There can be properties composed by other sub-properties. For example, the 'position' property may have three sub-properties: 'latitude', 'longitude' and 'altitude'.

Attributes:

- propertyName: A name that univocally identifies the property.
- vocabulary: The vocabulary that defines this property.
- isComposite: Indicates whether the property is composed by other subproperties.

Entity: Context Value

Description: This class represents a value for a Context Property. A Context Value might be associated to one or more Context Providers.

Attributes:

- value: This attribute conveys the value itself.

Entity: Context Property Description

Description: It represents the description of a Context Property and incorporates different attributes for conveying such metadata.

Attributes:

- datatype: The datatype of the property.
- measuredQuality: This attribute represents the quality that a property may represent. For example, ‘length’ for the 'width' property.

Entity: Context Value Description

Description: It represents the description of the value of a Context Property and incorporates different attributes for conveying such metadata.

Attributes:

- validityPeriod: Indicates the period of time (in seconds) for which this property value will be valid. After that period (counted from the timestamp), the value will be considered as out of date.
- timeStamp: a timestamp that indicates the precise moment in time when the property value was obtained.
- measurementUnit: The unit that might correspond to the value. For example, "kilo" for a "weight" property value.
- confidenceLevel: This property indicates the level of confidence for the property value.

Entity: Context Provider

Description: It represents a source of context information i.e. it is capable of providing the values of Context Properties.

Attributes:

- providerId: An identifier for the provider

Association Description

- **subproperty**: This association links a Property with its subProperties.
- **derivedProperty:** This association links a property with the properties from which it derives. A Property A derives from another Property B if the value of A depends on the value of B. This dependency is typically captured as some kind of computation.

Context of Use Meta-Model

This meta-model captures the context of use in which a user is interacting with a particular computing platform in a given physical environment in order to achieve an interactive task.

The figure below is a UML representation of this meta-model.

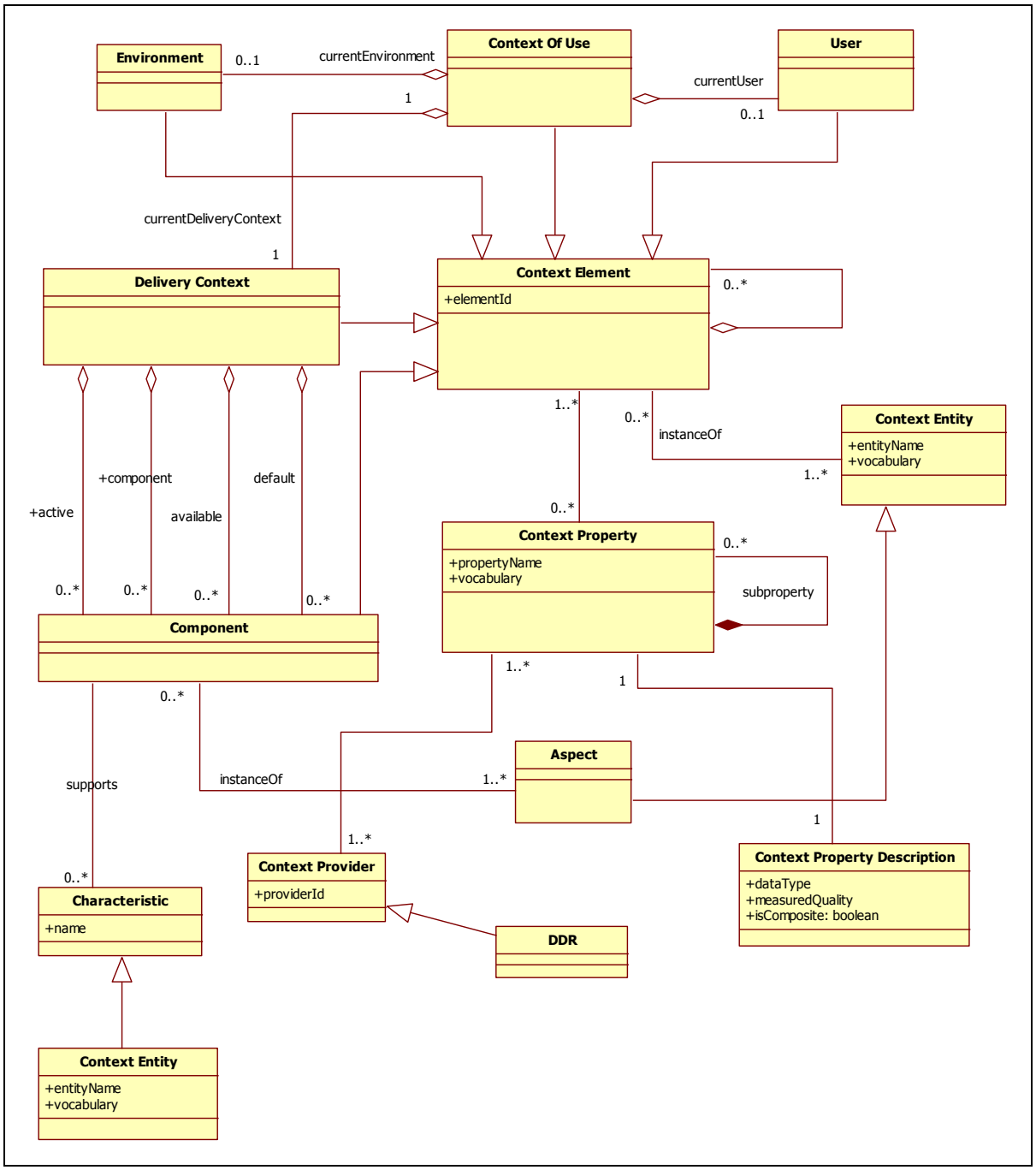

Figure 6 Context of use Meta-Model diagram

Entity: Context Provider

Description: It represents a source of context information i.e. it is capable of providing the values of Context Properties.

Attributes:

- providerId: An identifier for the provider.

Entity: Context Of Use

Description: This class represents the whole Context of Use as a Context Element.

Entity: Environment

Description: This class is a model of the physical environment where the interaction takes place.

Entity: User

Description: This class is a model of the human being who is currently perceiving and interacting with the system.

Entity: Delivery Context

Description: This class encompasses the set of hardware, software and network platform(s), that is, the computational, interaction and communication Component(s) that are used for perceiving and interacting with the system.

Entity: Aspect

Description: It represents a class of physical or software resource that can be part of the Delivery Context and which participates in delivering a user experience. Examples or Aspects are 'Camera', 'Device', 'Web Browser', etc.

Entity: Component

Description: A Component is an instance of an Aspect. In any Delivery Context an Aspect may have zero or more Components. For example, some devices have a primary and a secondary camera.

Entity: Characteristic

Description: This class models a Context Entity which represents a feature supported by a Component of the Delivery Context. Example of Characteristics are image formats (gif, jpeg ...), markup languages or fonts.

Attributes:

- name: The name of the characteristic.

Entity: DDR

Description: This class represents a Device Description Repository, aspecialization of Context Provider. A Device Description Repository is a repository that contains a priori known information that describes the characteristics and behaviours of the Components that may be part of a Delivery Context.

Association Description

- **active**: This association links a Delivery Context with the Components which are currently active. A Component is active if it is currently participating in delivering a user experience.
- **available**: This association links a Delivery Context with the Components which are currently available. A Component is available if it is ready to be activated by the user.
- **component**: This association links a Delivery Context with its Components.
- **default**: This association links a Delivery Context with its default Components. A Component is marked as a default Component if it will be used in the absence of any other explicit preference from the user.
- **supports:** It links a Component of the Delivery Context with its supported characteristic

Delivery Context Model

This model describes the characteristics (and situation) of the hardware, software and network platform(s), that is, the computational, interaction and communication Components that are used for perceiving and interacting with a system.

The Delivery Context Model proposed by WP1 is based on the “Delivery Context Ontology” W3C Working Draft [CAN09], which editors are the same authors as this deliverable.

The **rationale** for selecting such a specification lies on the fact that it represents a general consensus about how to represent the Delivery Context. The standardization process began on 2007 and the referenced draft has been published as a W3C Working Draft (16 June 2009). In addition such specification is widely supported by different first tier organizations, such as Telco Operators (Vodafone, Telefónica, Telecom Italia, AT&T), independent software vendors (Mobileaware, Volantis …), big software vendors (IBM) and Academic Institutions (UCL, ISTI-CNR).

The figure below outlines the main entities that are modelled by the Delivery Context Ontology. A thorough description of the Delivery Context Ontology can be found on the corresponding W3C Last Call Working draft. Additional diagrams will be provided in future versions of this document and in additional W3C WG Note that it is being created by the same authors as this deliverable.

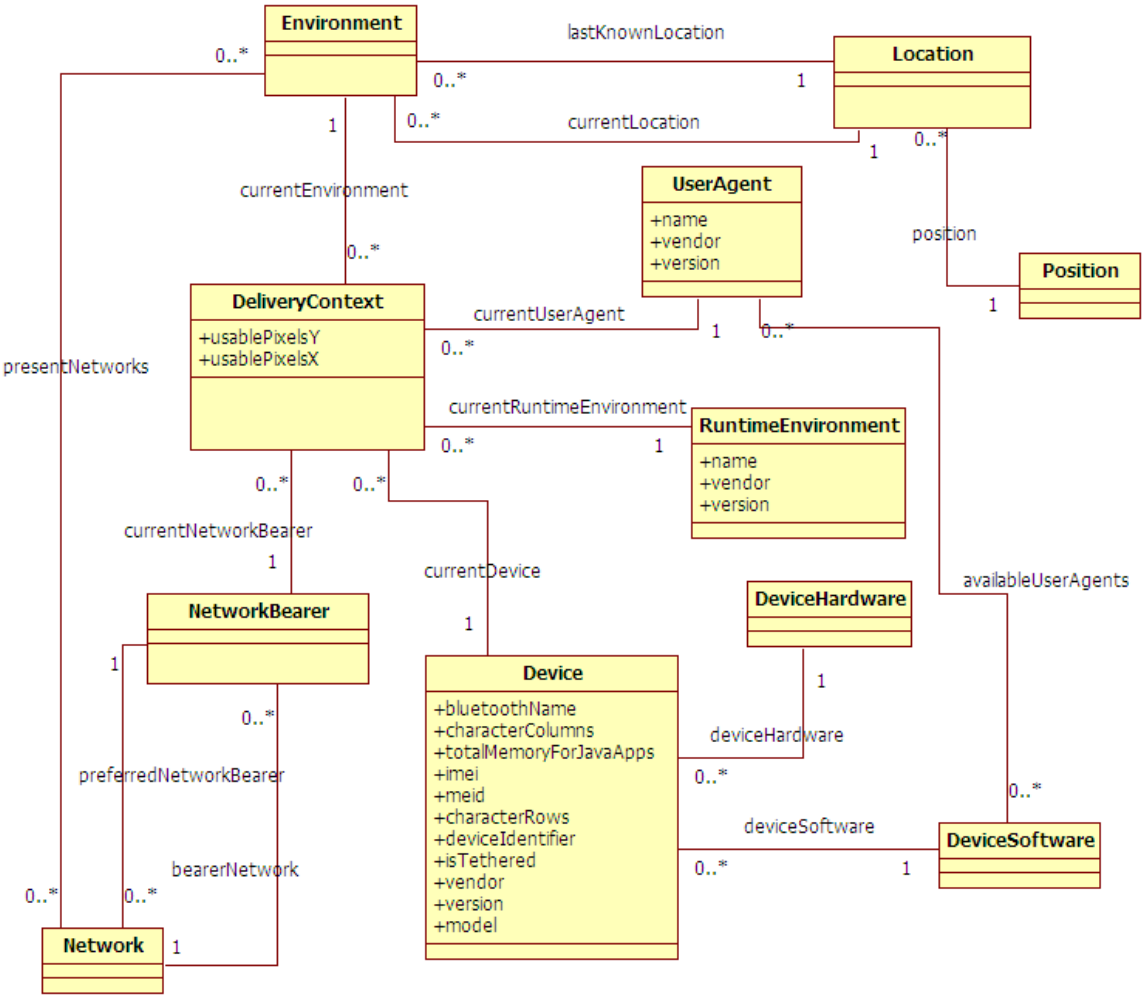

Figure 7 Delivery Context Model

1.2.5 Domain Model

A Domain Model is proposed below. Such domain model is independent of the technology or representation mechanisms used for maintaining information in an interactive system.

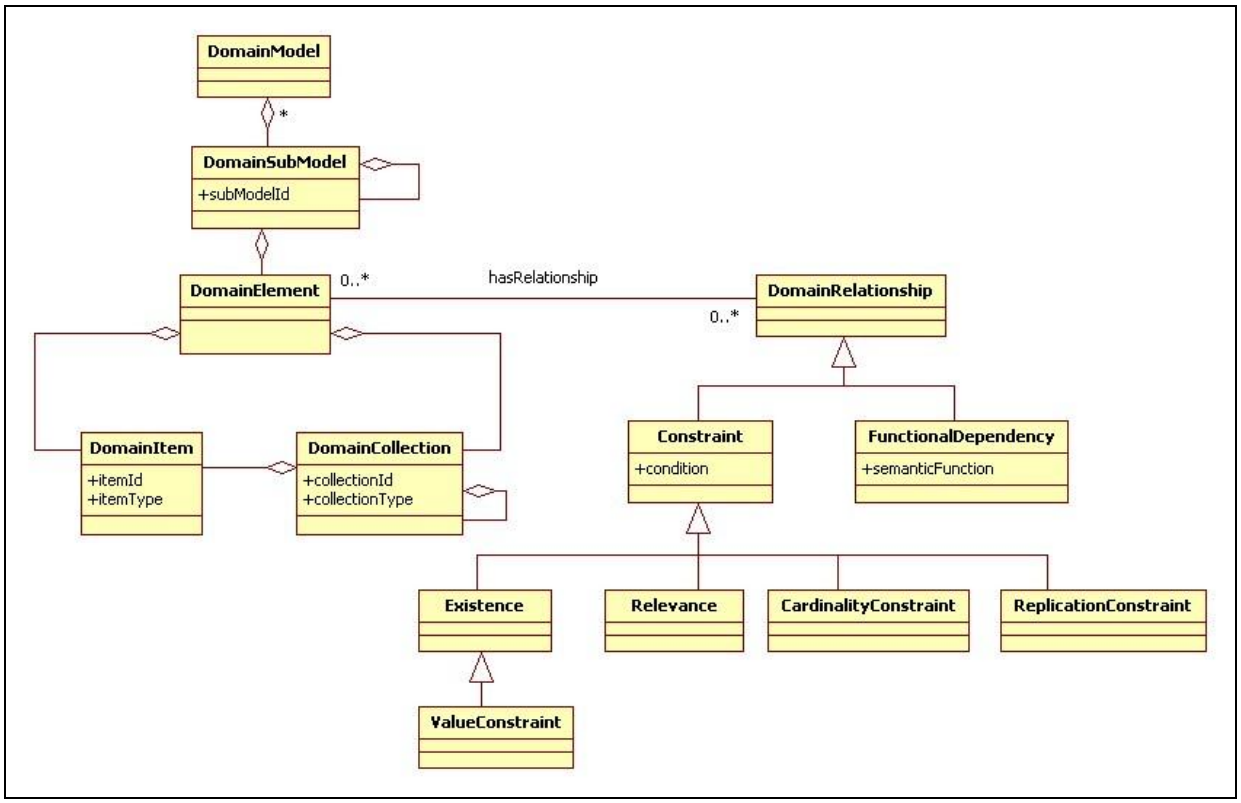

Figure 8 Meta-model of the NEXOF-RA Domain Model

Entity: DomainModel

Description: Domain model is a description of the classes of objects manipulated by a user while interacting with a system. The domain model is composed of submodels.

Entity: DomainSubModel

Description: This element provides a way to represent a domain model that at the same time can be composed of a different domain submodel. For instance, in XForms different submodels can be within a model and they can be associated to different domain facets, it is similar than the concept of package in Java.

Attributes:

- subModelId: Is an identifier of the DomainSubModel among all DomainSubModels. As such the id is unique.

Entity: DomainElement

Description: Consists of DomainItems. Consists of DomainCollections.

Entity: DomainItem

Description:

Attributes:

- itemId: Is an identifier of the DomainItem among all DomainItems. As such the id is unique.
- itemType: Type of item.

Entity: DomainCollection

Description: Express the notion of data collection.

Consists of DomainCollections. A concrete example of a collection of collections is a table. Consists of DomainItems. A concrete example of a collection of items is a comboBox.

Attributes:

- collectionId: Is an identifier of the DomainCollection among all DomainCollections. As such the id is unique.
- collectionType: Type of collection.

Entity: DomainRelationship

Description: Describes various types of relationships between domain elements. They can be classified in two types: Constraint and FunctionalDependency.

Entity: FunctionalDependency

Description:

Attributes:

- semanticFunction:

Entity: Constraint

Description: A constraint is an affection to domain elements.

Attributes:

- condition: Enables to specify a condition attached to a constraint. A condition is a rule that must be fulfilled in the specification before or after the application of a constraint.

Entity: Existence

Description: Denotes the existence affection to domain elements. One domain element needs the existence of other domain elements.

Entity: ValueConstraint

Description: Denotes the existence affection to domain elements. One domain element needs the existence of a value in other domain element.

Entity: Relevance

Description: Denotes the relevance affection to domain elements. One domain element is more relevant than other domain elements.

Entity: CardinalityConstraint

Description: Denotes the cardinality affection to domain elements.

Entity: ReplicationConstraint

Description: Denotes the replication affection to domain elements. One element is replicated.

1.2.6 Task Model

A task model is a description of the tasks that a user will be able to accomplish in interaction with the system. This description is a hierarchical decomposition of a global task, with constraints expressed on and between the subtasks [FLO06].

The proposed task model has been contributed by the UsiXML project (under the Open Construction Process) [USIXML] and it is an (slightly) extended version of ConcurTaskTree (CTT) [PAT03]: a hierarchical task structure, with temporal relationships specified between sibling tasks.

The UML diagram below illustrates such a task model. The description of the entities and relationships follows the diagram.

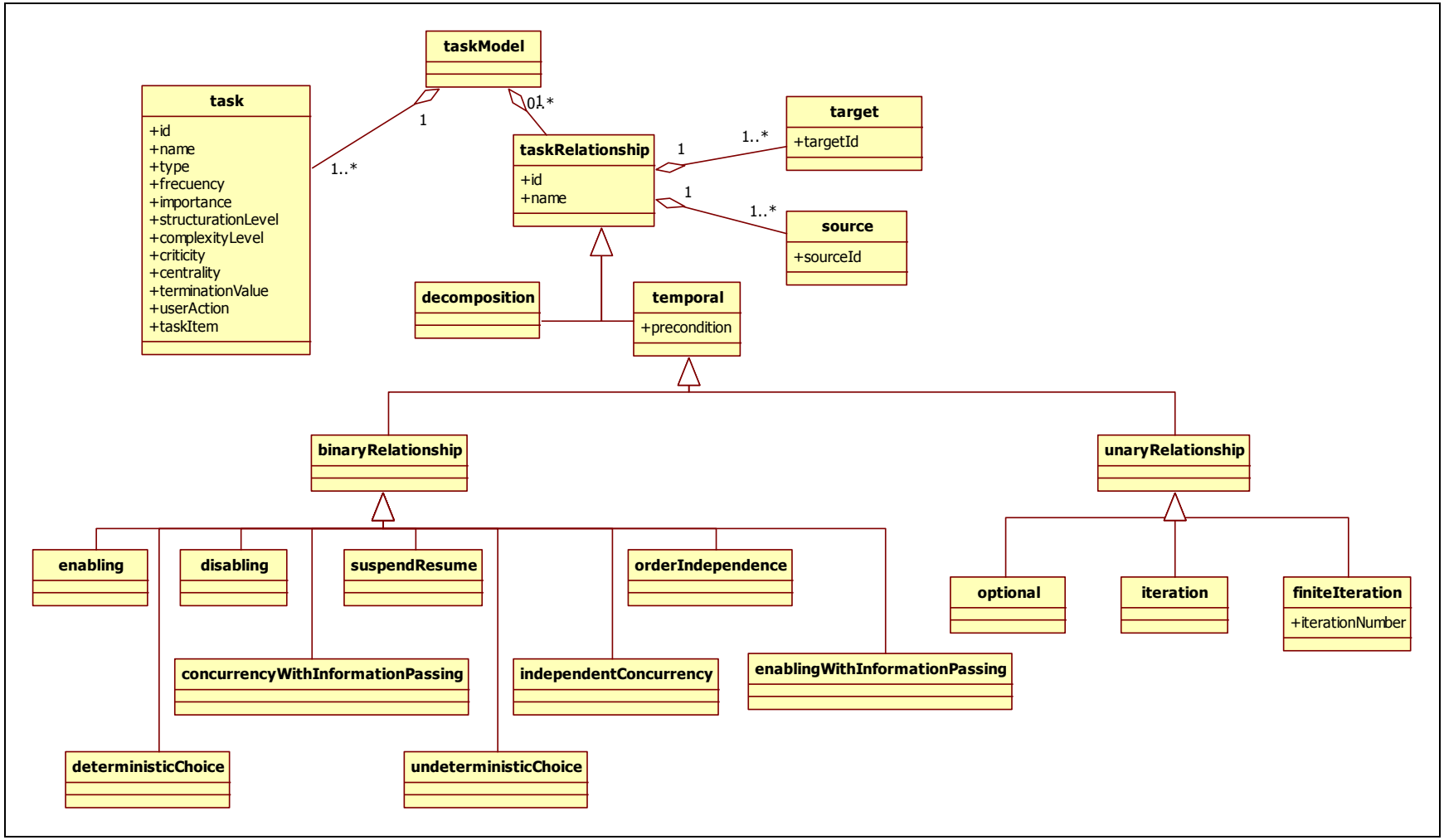

Figure 9 Meta-model of the NEXOF-RA Task Model

Entity: Task

Description: Task is the basic structure that composes the task model. Tasks are activities that have to be performed to reach a goal.

Attributes:

- id: It is the identifier of the task.
- name: It is the name of the task.

Entity: TaskRelationship

Description: Tasks are linked by two types of relationships (taskRelationship): Hierarchical relationships (decomposition) and temporal relationships (temporal).

Attributes:

- id: It is the identifier of the taskRelationship.
- name: It is the name of the taskRelationship.

Entity: Target

Description: It represents the target of a task relationship.

Attributes:

- targetId: It is the identifier of the target task.

Entity: Source

Description: It represents the source of a task relationship.

Attributes:

- sourceId: It is the identifier of the source task.

Entity: Decomposition

Description: Each task can be decomposed into two or more subtasks.

Thus, with the exception of the root task, each task has a mother task from which the temporal relationships are inherited.

Entity: Temporal

Description: Temporal relationships between the tasks are specified with temporal operators.

Temporal relationships are of two types: unary and binary.

Entity: BinaryRelationship

Description: Binary relationship link together two tasks.

Entity: Enabling

Definition: Enabling relationships specify that a target task cannot begin until source task is finished.

Entity: Disabling

Definition: Disabling relationships refer to source task that is completely interrupted by a target task.

Entity: SuspendResume

Definition: Suspend Resume relationships refer to source task that can be partially interrupted by a target task and after the target task is completed the source task will be concluded.

Entity: OrderIndependence

Definition: Order Independence relationships are when two tasks are independent of the order of execution.

Entity: ConcurrencyWithInformationPassing

Definition: Concurrency with Information Passing relationships are a type of temporal relationships where two tasks are in concurrency execution and passing information between them.

Entity: IndependentConcurrency

Definition: Independent Concurrency relationships are a type of temporal relationships where two tasks are executed concurrency but are independent one to each other and there is no information interchange.

Entity: EnablingWithInformationPassing

Definition: Enabling with Information Passing relationships specify that a target task cannot be performed until the source task is performed, and that information produced by the source task is used as an input for the target task.

Entity: DeterministicChoice

Definition: Deterministic Choice relationships refer to two source tasks that could be executed but once that one task is initiated the other cannot be accomplished anymore.

Entity: UndeterministicChoice

Definition: Undeterministic Choice relationships define the relation between two source tasks in which both task could be started but once one task is finished the other cannot be accomplished anymore.

1.2.7 Aui Model

An Abstract User Interface Model is an expression of the rendering of the domain concepts and tasks in a way that is independent from any modality of interaction [FLO06].

We propose an *Aui Model* represented in the UML diagram below. Afterwards a description of the main entities and relationships is provided.

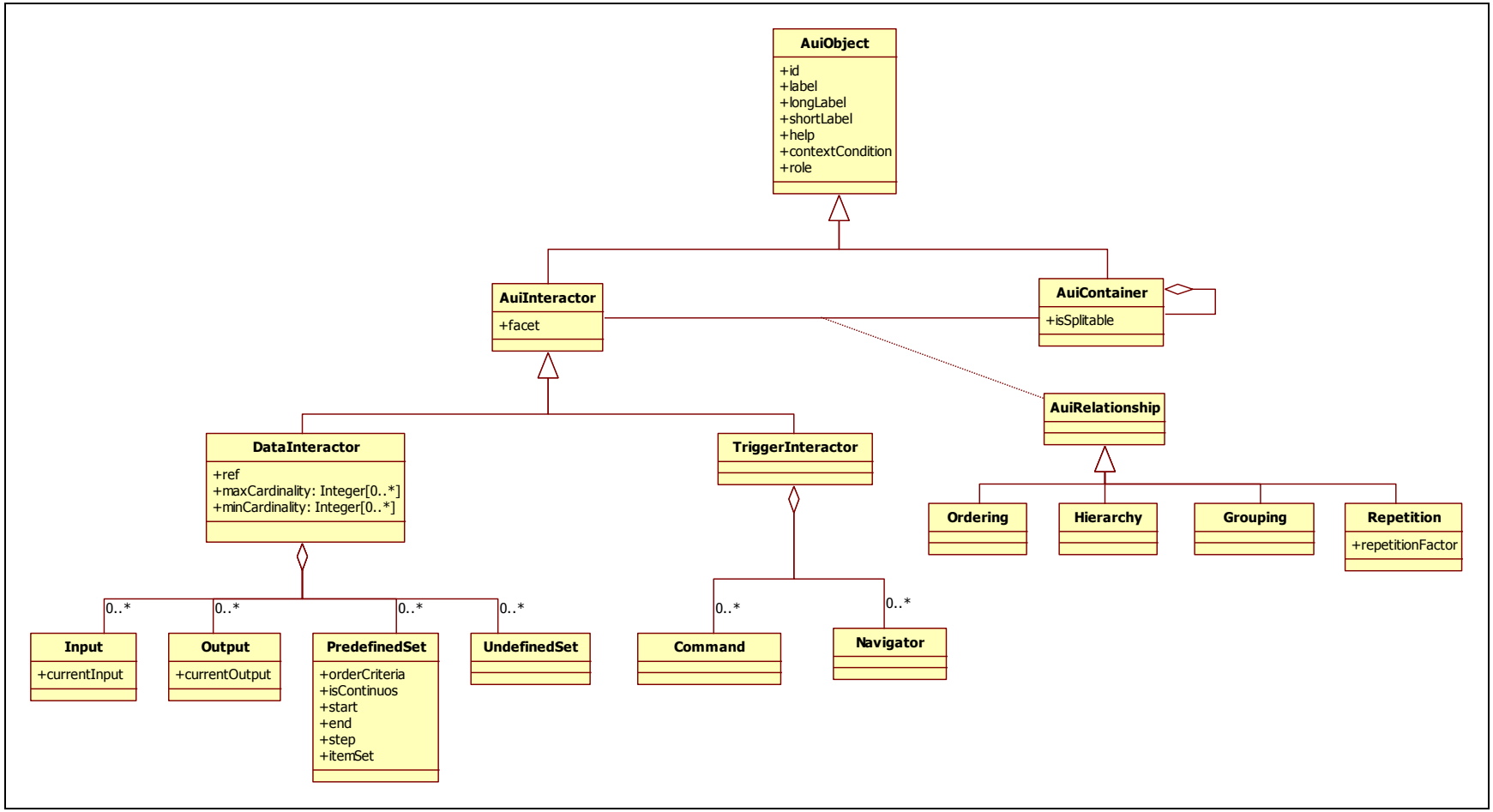

Figure 10 Meta-Model of the AUI Model for NEXOF-RA

Entity: AuiObject (Abstract User Interface Object)

Description: This the more general concept situated at the root of the hierarchy. AuiObjects are the elements populating the AUI Model.

Attributes:

- id: It is a machine-readable unique identifier
- label: A human-readable label for the object
- longlabel: A long description of the object
- shortlabel: A concise description of the object
- help: A help string to assist users
- contextCondition: A expression to be evaluated in order to check if the object is relevant under the current Context of Use or not
- role: This attribute allows to assign high level semantics to an AuiObject. (It is similar to the XHTML Role Attribute Module [XHTML Role Attribute Module])

Entity: AuiContainer

Description: This object is a Container for grouping elements. It defines groups of tasks that have to be presented together, in the same window or page, for example. The container could contain both AuiInteractor elements or other AuiContainer elements.

The relationship between the AuiContainer and the AuiInteractors is declared using an AuiRelationship object.

Attributes:

- isSplitable: Indicates whether the container could be splitted in two or more parts, when presenting it to the user. (For instance to allow pagination)

Entity: AuiInteractor

Description: AuiInteractors are individual elements populating an abstract container

Entity: DataInteractor

Description: They are AuiInteractors aimed to present and / or obtain data to and from the user. They are represented as an aggregation of different facets (Input, Output, PredefinedSet, UndefinedSet) describing the type of interactive tasks they are able to support.

Attributes:

- ref: It is a reference to the domain model element.
- maxCardinality: It represents the maximum cardinality for the element.

- minCardinality: It represents the minimum cardinality for the element.

Entity: Input

Description: This entity represents the input facet of a DataInteractor, which indicates that such interactor accepts the input of information from the user.

Attributes:

- currentInput: It represents the current inputted value.

Entity: Output

Description: It represents the facet of a DataInteractor aimed to present information to the user.

Attributes:

- currentOutput: It will contain the current output value for the element.

Entity: PredefinedSet

Description: This entity is the representation of the facet of a DataInteractor that serves for the purpose of entering or displaying information coming from a set of predefined items.

Attributes:

- orderCriteria: It declares whether the values are ordered
- isContinuos: It declares whether the values are continuous
- start: Indicates the initial value for the set
- end: Indicates the final value for the set
- step: Indicates the difference between 2 contiguous values of the set
- itemSet: It contains the set of values

Entity: UndefinedSet

Description:. This entity is the representation of the facet of a DataInteractor intended to enter or to display information that it is not a priori known.

Entity: TriggerInteractor

Description: It is an interactor that allows users to give orders to the system or to navigate to the functionalities they may need in a given moment. It can have two different facets: Command and Navigator

---

Entity: Command

Description: This entity represents a facet of a TriggerInteractor that serves for the purpose of command the system to perform a computation

---

Entity: Navigator

Description: It is a TriggerInteractor facet aimed to change the current presentation unit, thus allowing users to change to a different set of tasks to be accomplished.

Entity: AuiRelationship

Description: This is an association class aimed to indicate explicitly what is the exact relationship between a AuiContainer and its contained AuiInteractors

---

Entity: Ordering

Description: This relationship denotes that there is an order relationship between the contained elements

---

Entity: Hierarchy

Description: This relationship indicates that the contained elements form a hierarchy.

Entity: Grouping

Description: It declares a relationship between two or more AuiInteractors of the same AuiContainer that need to be grouped together, regardless of the actual layout that will be defined at the Concrete User Interface model.

Entity: Repetition

Description: This entity is intended to represent that the AuiObjects present in the AuiContainer will be repeated a certain number of times controlled by the domain model

1.3 The Composition Viewpoint: A User-Oriented Composition Model for Service Front-Ends

The figure below is a UML diagram that represents the EzWeb model for user-oriented composition of personalized user interfaces and front-ends for services. This diagram is followed by the description of the entities involved and their relationships.

This diagram is the result of the contribution made by the EzWeb NESSI Statregic Project.

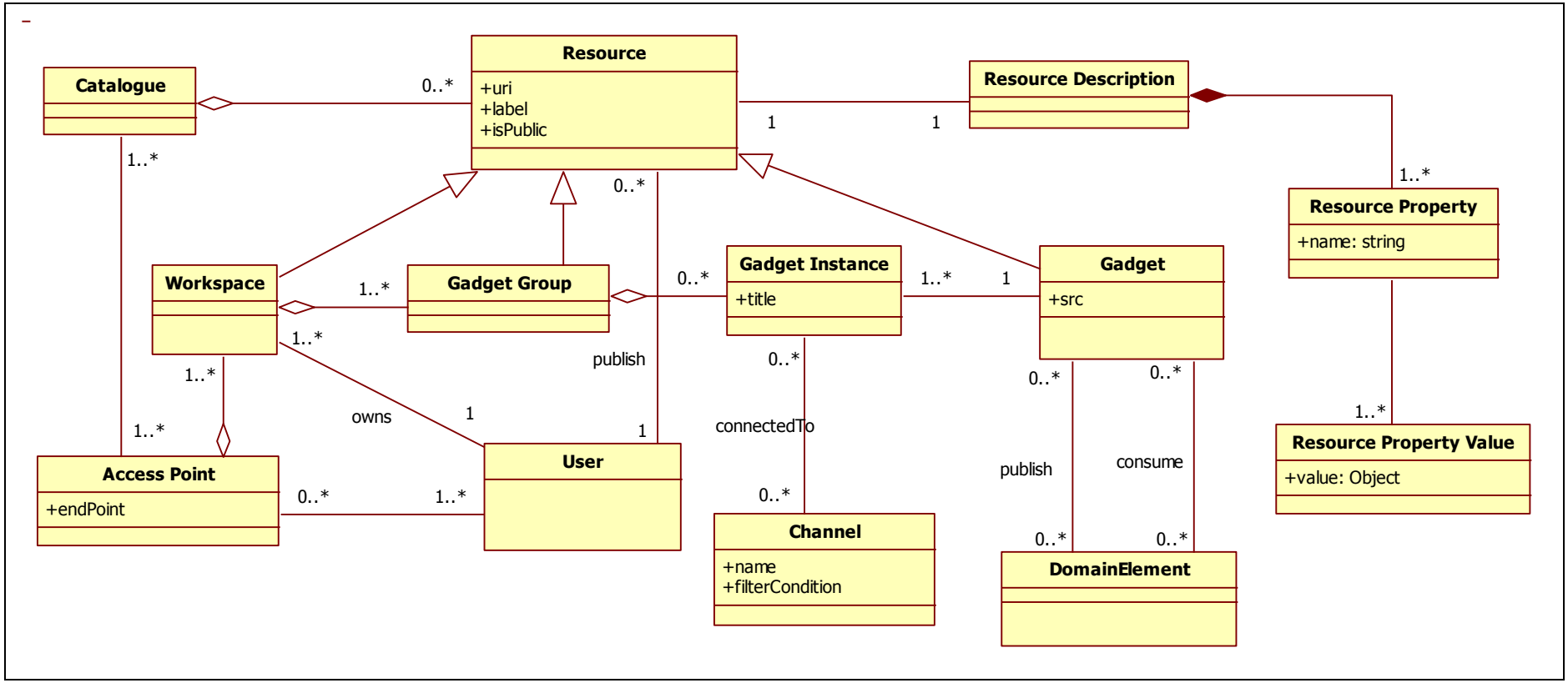

Figure 11 User-Oriented Composition Model for Service Front-Ends

Entity: Resource

Description: It represents any entity that can be used to compose a Service Front-End. Resources are identified by URIs and are indexed in a Catalogue.

Attribute:

- uri: Uniform Resource Identifier of the resource
- label: Label for the resource
- isPublic: Indicates whether this resource is publicly available or not.

Entity: Resource Description

Description: This class represents all the metadata that serves to describe a Resource. A description is an aggregation of different Resource Properties.

Entity: Resource Property

Description: This class represents a property of a Resource. Each property is identified by a name. A Resource Property can have zero or more values.

Entity: Resource Property Value

Description: This class conveys a value of the ResourceProperty.

Entity: Catalogue

Description: A repository devoted to store meta-information about Resources.

Entity: Access Point

Description: It represents the main front-end utilized by users to create their own composite user interfaces. Hence, it provides access to one or more user Workspaces indexed in one ore more Catalogues.

Entity: Workspace

Description: A Resource intended to contain composite service front-ends composed by several interconnected Gadgets.

Entity: Gadget Group

Description: This class represents a group of gadgets that have been composed in the same presentation unit of a workspace. A workspace may have several Gadget Groups and each Gadget Group is bounded to a Workspace.

Entity: Gadget

Description: A resource that implements the user interface and application logic necessary to interact with one or more underlying services. Gadgets publish events to, and consume events from, other Gadgets.

Entity: Gadget Instance

Description: It is a concrete instance of a Gadget. A Gadget Instance is bounded to a Gadget Group

Entity: Channel

Description: A Channel is a communication entity capable of routing data between Gadget instances. A Gadget Instance can be connected to one or more channels.

Entity: Domain Element (See the Domain Model)

Description: A Gadget can publish or consume different Domain Elements

Entity: User

Description: This is a representation of the end-user. A end-user may publish zero or more Resources, may own one or more workspaces and may have access to zero or more Access Points.

1.4 Information Models

1.4.1.1 Information Model for User Interface Composition

The model is described in the UML diagram depicted below. As the reader may have guessed this information model is closely related to the User-Interface Composition Model described on section 1.3

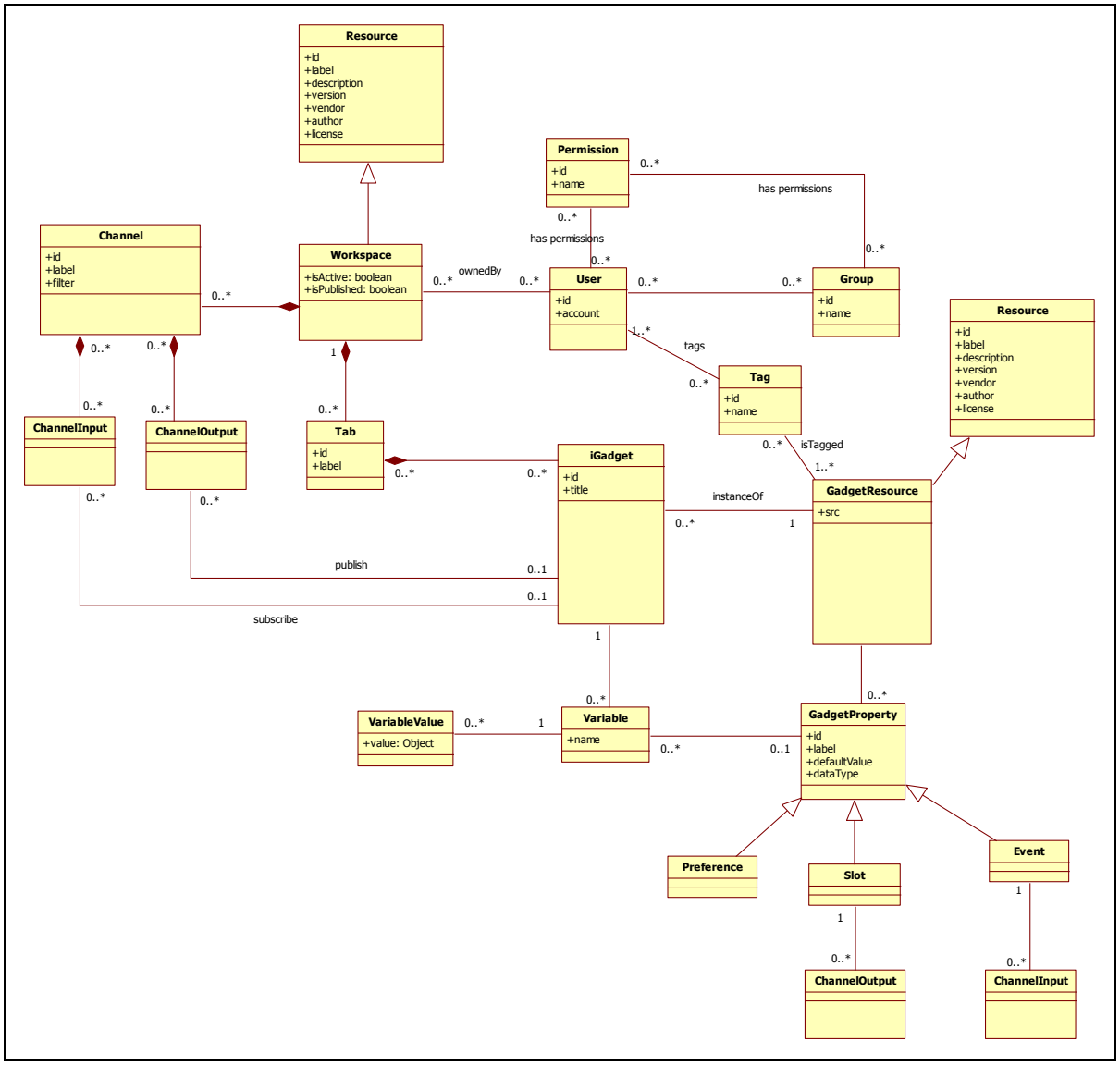

Figure 12 Information Model

Entity: Resource

Description: Any of the elements present in the front-end, could be a workspace, a gadget, a tab…

Entity: Workspace

Description: It is the resource that groups the tabs and the channel that the user combined to solve a domain problem. Could be active or not, and could be published or not.

Entity: Tab

Description: It is a group gadget instances that has been composed in a workspace. A workspace is composed by several Tabs (Gadget Groups).

Entity: Channel

Description: It is an aggregation of ChannelInput and ChannelOutputs allowing gadgets to communicate data between them.

Entity: ChannelInput

Description: It represents the information required by a gadget as an input.

Entity: ChannelOutput

Description: It represents the information provided by a gadget as an output.

Entity: User

Description: It is the representation of the user of the system. The user could belong to groups, and could have different permissions.

Entity: Permission

Description: The user is allowed to do some actions, represented by permissions.

Entity: Group

Description: It is a collection of users with common characteristics. A group is associated with a collection of permissions.

Entity: iGadget

Description: It is a concrete instance of a Gadget. A Gadget can have many instances

Entity: Gadget Resource

Description: A resource that implements the user interface and application logic necessary to interact with one or more underlying services. They publish events to, and consume events from, other Resources.

Entity: Tag

Description: A tag is a concept that categorizes a GadgetResource. They are useful for searching purposes.

Entity: GadgetProperty

Description: It denotes an information item that a Gadget can publish or consume. The gadget property contains the name of the property, the default value and the data type.

Entity: Variable

Description: It contains the name of a variable related to an instance of a gadget. A Variable is the mechanism used by a Gadget Instance (iGadget) for actually publishing or consuming information.

Entity: VariableValue

Description: It contains the value of a variable related to a gadget instance.

Entity: Preference

Description: It represents a Gadget Preference, understood as user-provided configuration data.

Entity: Slot

Description: A Slot is a subclass of Gadget Property that denotes an information item that a Gadget can consume. It could be connected to a ChannelOutput in order to receive the actual values (from other gadgets).

Entity: Event

Description: A Slot is a subclass of Gadget Property that denotes an information item that a Gadget can publish. It could be connected to a ChannelInput in order to provide the actual values (to other gadgets)..

REFERENCES

[CAL03]

Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L. and Vanderdonckt, J. (2003), A Unifying Reference Framework for Multi-Target User Interfaces, Interacting with Computers, V15N3, June, 289-308. (See http://www.isys.ucl.ac.be/bchi/publications/2003/Calvary-IwC2003.pdf)

[CAN09]

Jose Manuel Cantera, Rhys Lewis , Delivery Context Ontology, W3C Editor's Draft 02 March 2009. (See http://www.w3.org/2007/uwa/editors-drafts/DeliveryContextOntology/LastCallWD-March2009)

[CRO02]

Crowley, J., Coutaz, J., Rey, G., Reignier, P., 2002. Perceptual Components for Context- Aware Computing. Proceedings of International Conference on Ubiquitous Computing UbiComp'2002 (Göteborg, 29 September-October 1 2002). Lecture Notes in Computer Science Vol. 2498, Springer Verlag, Berlin, pp. 117–134.

[DEY01]

Anind K. Dey, Understanding and Using Context, Personal and Ubiquitous, volume 5, number 1, year 2001, pages 4-7.

[FLO06]

Murielle Florins, Graceful Degradation: a Method for Designing Multiplatform Graphical User Interfaces, Ph.D. thesis, Université catholique de Louvain, Louvain-la-Neuve, Belgium, 11 July 2006.

http://www.isys.ucl.ac.be/bchi/publications/Ph.D.Theses/Florins-PhD2006.pdf

[MariaXML]

Paternò F.; Santoro C.; Spano, L. D.: Designing Usable Applications based on Web Services; I-USED'08, September 24, 2008, Pisa, Italy

http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-407/paper15.pdf

[PAT03]

Paternò, F., Santoro, C., 2003. A Unified Method for Designing Interactive Systems Adaptable To Mobile And Stationary Platforms. Interacting with Computers, in this volume.

[RAG09]

Dave Raggett, Model-based User Interfaces Incubator Group Charter, W3C Incubator activity, 26 October 2009.

[SEF04]

Seffah, A., & Javahery, H. (2004). Multiple user Interfaces: Cross-Platform Applications and Context- Aware Interfaces. In A. Seffah & H. Javahery (Eds.), Multiple User Interfaces: Engineering and Application Framework (pp. 11-26). Chichester (GB): Wiley and Sons.

[USIXML]

UsiXML. User Interface eXtensible Markup Language. (See http://www.usixml.org/)

[W3C04]

W3C. Device Independent Working Group. (See http://www.w3.org/2004/05/di-charter-2004-06.html)

[W3C08]

W3C. (2008). Model-based User Interfaces Incubator Group Charter (See http://www.w3.org/2005/Incubator/model-based-ui/charter/)

[W3C09]

W3C. (2009). Accessible Rich Internet Applications (WAI-ARIA) 1.0. W3C Working Draft 24 February 2009. (See http://www.w3.org/TR/2009/WD-wai-aria-20090224/)

[W3C07]

W3C. (2007) Ubiquitous web applications activity (See http://www.w3.org/2007/uwa/)