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

Project no. FP7 – ICT – 258030

Deliverable D2.3.2
CARFO & CARFO Population(R2)

Due date of deliverable: 28/02/2013

Actual submission to EC date: 28/02/2013

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

Document Information	
Lead Contractor	
Editor	Fundación CTIC
Revision	v.1.0 (25/02/2013)
Reviewer 1	UCL
Reviewer 2	
Approved by	
Project Officer	Mr. Michel Lacroix

Contributors	
Partner	**Contributors**
CTIC	**Carlos Tejo-Alonso, Emilio Rubiera, Luis Polo Paredes, Cristina González Cachón**
TID	**Javier Caminero, Mari Carmen Rodríguez-Gancedo**
SAP	**Safdar Ali**
UCL	**Vivian Motti**
W3C	**Dave Raggett**

Changes			
Version	**Date**	**Author**	**Comments**
0.1	17/01/2013	CTIC	Table of contents from Wiki
0.2	29/01/2013	CTIC	First version
0.3	08/02/2013	TID, CTIC, UCL, W3C, SAP	First contributions provided by partners
0.4	14/02/2013	CTIC	Consolidation sent to partners and reviewers
0.5	21/02/2013	CTIC	Reviewed version by UCL
1.0	25/02/2013	CTIC	Final version

Executive Summary

This deliverable presents the final results of work undertaken in the FP7 SERENOA project in the topic of semantic knowledge representation for Service Front End adaptation. In it, we describe the overall fit of such a system in the general picture of the SERENOA architecture and runtime and discuss several topics related to the design and contents of the ontology itself, the ontology population, publication and access.

We present the final version of the CARFO Ontology in the shape of a module for Context of Use in the SFE adaptation domain. CARFO is expressed using the W3C standard ontology language OWL2 and includes the global formalization of concepts relative to the User, the Platform, the Environment and some other aspects related to the content, namely structure and presentation.

Users are a central pillar of the CARFO ontology, as interface adaptation must consider not only physical and circumstantial aspects but user's preferences and profiles of target applications. In this sense, CARFO reuses some ontologies already available to describe several aspects of users.

The second pillar of the CARFO ontology is the “platform”, which comprises all the elements of hardware, software and networks relevant to describe a computing environment. This is a much broader concept than just a server or client side framework, such as MyMobileWeb or Android. Therefore a comprehensive description of the platform is challenging, because it involves different abstraction levels, a wide range of devices and complex software applications, such as web browsers and operating systems.

The third pillar of the context of use module in CARFO is the "environment" surrounding the user and the platform. In CARFO, the environment is interpreted in a broad sense, comprising not only ambient conditions but also contextual ones. The environment would be then the information or knowledge that surrounds an entity, physical or conceptual. The information about the environment is infinite, but the relevant context of an entity is finite and enclosed.

The ontology population consists in filling up a given ontology by providing domain instances (i.e., RDF datasets) and relationships described according to its semantic model, being thus the result of a linear workflow featuring steps as the identification of information sources, selection, transformation, enrichment and storage. Three sources have been used to provide the population of the CARFO ontology: caniuse.com, Nokia Developer site and CTIC Device Description Repository (DDR).

Finally, the ontology publication includes the definition of an ontology namespace, i.e., the URI that identifies the ontology. It is also necessary to host the ontology files in a public repository and associate the URI with this hosting location. The official name of the ontology has been decided by the consortium to be CARFO (CARF Ontology), as it is a formalization of the CARF framework. A namespace has been registered at PURL for the ontology: http://purl.org/carfo. The ontology is hosted by CTIC in http://vocab.ctic.es and is published according to W3C best practices. All the data is available from an SPARQL endpoint: http://data.ctic.es/sparql at the named graph http://purl.org/carfo/example.

Table of Contents

1 Introduction

This deliverable presents the final results of work undertaken in the FP7 SERENOA project in the topic of semantic knowledge representation for Service Front End adaptation. In it, we describe the overall fit of such a system in the general picture of the SERENOA architecture and runtime and discuss several topics related to the design and contents of the ontology itself, the ontology population, publication and access.

We present the final version of the CARFO Ontology in the shape of a module for Context of Use in the SFE adaptation domain. This includes the global formalization of concepts relative to the User, the Platform, the Environment and some other aspects related to the content, namely structure and presentation. This deliverable is integrated by a merge of the initially expected deliverables D2.2.2 CARFO (R2) and D2.3.2 CARFO Population (R2) and new contents.

1.1 Objectives

The objectives of this deliverable are:

- The description of the CARFO ontology: ontological assumptions, design decisions and main classes and properties.
- A summary of the ontology population process, which is automatically carried out from several data sources: caniuse.com, CTIC Device Description Repository (DDR) and Nokia Developer site.
- The publication and access of the CARFO ontology and the available dataset obtained from the population process.

1.2 Audience

The intended audience of this document is fourfold:

a) First and foremost, the members of the consortium, who may find here a detailed account of what this part of the SERENOA environment is. The members involved in the development of the CARF (Context-Aware Reference Framework), CADS (Context-Aware Design Space), and CARFO ontology should refer to this document as an insight into the use of these modules and their interaction (e.g. interfaces) with the CARFO knowledge base from within the SERENOA architecture
b) Secondly, developers that may find this document useful in order to get a short overview on how the CARFO ontology and CARFO knowledge base are used in SERENOA project to facilitate the Adaptation Server to make respective adaptations to the UIs according to the context information.
c) Thirdly, as a publicly available document, the researchers in the relevant fields: adaptation of SFEs, semantic technologies and also medium-scale project software engineering.
d) Last but not least, the EC officials that may use the information in this document as an account of the activities executed during the project tasks.

1.3 Related documents

The following SERENOA documents are related to the content of this deliverable:

- D1.1.2 Requirements Analysis (R2) – describes the requirements of the project in general and discusses the gathered requirements by means of CARFO ontology and knowledge Base.
- D1.2.1 Architectural Specifications (R1) provides useful indications about the project results that have been considered for the integration and use of CARFO Knowledge Base in the overall lifecycle of the SERENOA applications.
- D2.1.1 and D2.1.2 CARF and CADS (R1 and R2) – The two releases of the CARF (Reference Framework) and CADS (Design Space) for SFEs.
- D2.4.1 Criteria for the evaluation of CARFO ontology and Knowledge Base.
- D2.2.1 CARFO (R1)
- D2.3.1 CARFO population (R1)
- D3.1.1 Reference Models Specification (R1)

- D4.4.1 Context of Use Runtime Infrastructure
- D4.2.1 Algorithm Library for AAL (R1)
- D5.2.1 Applications Prototypes (R1) provides the details about the application

1.4 Organization of this document

This document is organized as follows. Section 2 describes the ontology. It provides a concise presentation of the main ontological design decisions inspired by the upper-level ontology: Dolce+Dns Ultralite. Furthermore, CARFO is grounded in well-known and spread vocabularies such as FOAF, FRAP, SIOC, etc. The population process is described in Section 3. This process is automatically performed from a selection of relevant knowledge bases. The output is an RDF dataset compliant with the CARFO ontology and publicly available through an SPARQL endpoint. Finally, Section 4 covers all the aspects related with the publication and access of the ontology and generated data.

2 Ontology definition

2.1 Introduction and general framework

In the context of SERENOA project, the CARFO ontology has been designed to capture information about the User, the Task, the Platform, the Environment, and/or various aspects of the content (structure and presentation). This increases the amount of contextual information that can be used to accomplish sophisticated adaptation. Moreover, current adaptive service front-end systems rely on their own formalism and vocabulary for data representation. By the use of CARFO as a standardized ontology, the systems can share and reuse model information to solve the inherent lack of data that hinders sophisticated adaptation. It provides formalism for the semantic annotation of the information transferred and reused across the systems.

The CARFO ontology serves two main purposes in SERENOA. The first is being the basis of a live, run-time available module, the CARFO Knowledge Base (CKB). This knowledge base might be used during execution by several of the modules in SERENOA to extract information about the adaption techniques, devices on which SERENOA may run, user information and so on. Thus, CARFO is not defined only as a purely formal instrument, but is also populated with the instances of knowledge that are needed to enable front-end dynamic adaptations.

Secondly, the ontology remains as the definitive 'dictionary' of concept definition for SERENOA. The ontology concepts have (partially) converged with those used for defining the elements in the AAL-DL, ASFE-DL and context description language, thereby using consistent definitions of concepts and relations throughout the project.

In all this process, we need to keep a necessary balance of complexity versus expressivity: if we possibly could fully define all concepts in adaptation and represent them in the ontology, maybe it would be excessively detailed for run-time usage and hence ultimately less useful. For that regard, many approaches and design decisions have been taken and documented in ontology documents.

CARFO uses some upper-level classes belonging to the DOLCE+DnS Ultralite (DUL) ontology. DUL is a simplification and an improvement of some parts of DOLCE Lite-Plus library[1] that provides a set of upper level concepts that can be the basis for easier interoperability among many middle and lower level ontologies. In this aspect, to model devices and their components, CARFO takes advantage of the DUL concept dul:PhysicalObject. To provide complete description of users, in terms of roles, profiles, communities and preferences, as it will be more specifically presented later, CARFO benefits from well-developed ontologies such as FOAF[2], WAI[3] and FRAP[4].

In addition, CARFO also reuses a number of properties that links its model to some standard vocabularies belonging to the European Commission initiative JoinUp[5], namely radion:keyword from the Repository Asset Distribution (RADion)[6], a model that facilitates federation and co-operation of systems and data by providing a common layer among repositories that want to exchange data, adms:relatedDocument and wdrs:describedBy from the Asset Description Metadata Schema (ADMS)[7], a metadata vocabulary to describe semantic interoperability assets.

2.2 Modular Design of CARFO

CARFO scope aims to cover a wide range of aspects, from description of interfaces to people and hardware. Due to this heterogeneity, SERENOA proposes a modular design of the ontology that permits to manage each module independently. Moreover, one of the goals is to reuse existing knowledge pieces in order to

[1] http://dolce.semanticweb.org
[2] http://foaf-project.org/
[3] http://purl.org/wai
[4] http://purl.org/frap
[5] http://joinup.ec.europa.eu
[6] http://www.w3.org/ns/radion
[7] http://joinup.ec.europa.eu/asset/adms/home

maximize interoperability.

OWL2 (Group, 2012) enables assembling knowledge from different ontologies, scattered in different files (or web locations). References can be made by URIs, or even by the owl:imports mechanism. In addition, OWL2 is represented as RDF graphs, which have the nice property to be seamlessly merged regardless of their origin. Figure 1 - Example of a modular ontology, illustrates both mechanisms to combine separated ontology modules in OWL2. A common pattern in this approach is to have a high-level ontology orchestrating several smaller ontologies.

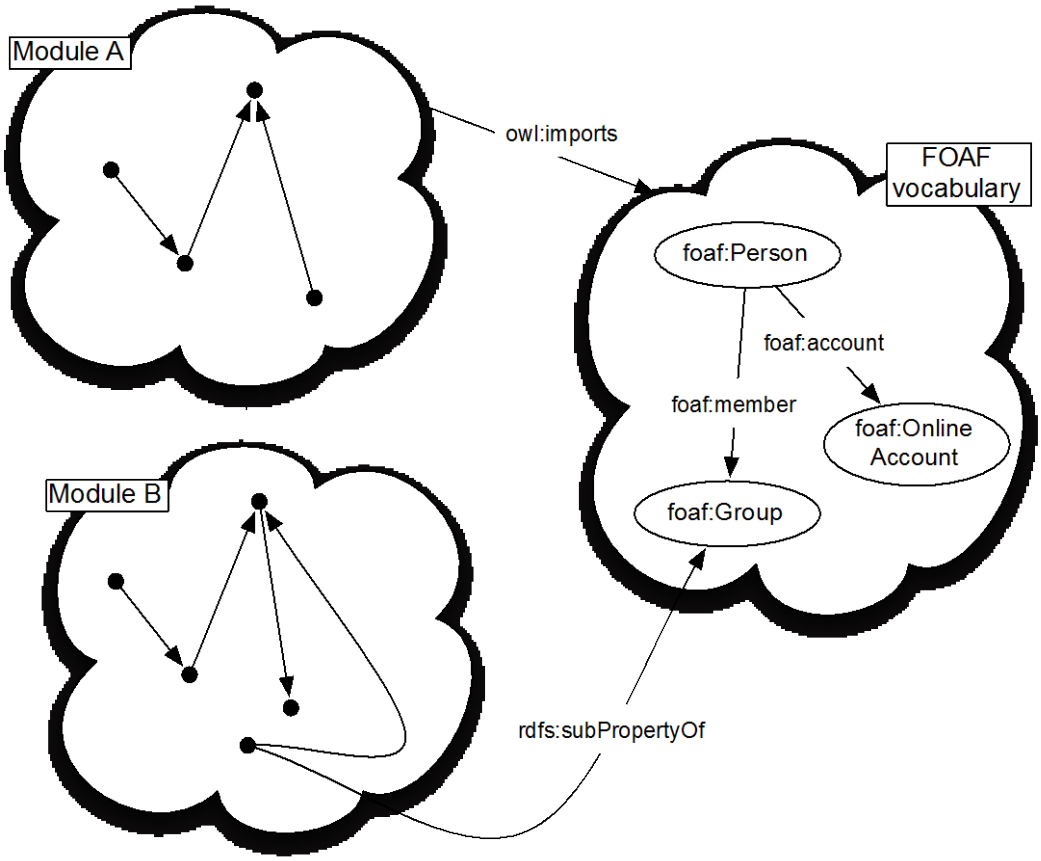

Figure 1 - Example of a modular ontology

A 1-to-1 mapping from CADS axes to CARFO modules has been discarded in early stages of the ontology development (see Deliverable D2.2.1). This is due to the different granularity of CADS axes. For instance, the Context of Use, an axis of CADS, is large enough to be considered as a set ontology schemas. On the other hand, the UI deployment axis, which indicates whether the deployment is static or dynamic, does not have the same complexity. As a consequence, the Context of Use has become the central pillar of the CARFO ontology. The rest of the axes are subsumed, if relevant, within its structure.

2.3 Context of use

2.3.1 User

Users are a central pillar of the CARFO ontology, as interface adaptation must not only consider physical and circumstantial aspects (the platform and environment) but users' preferences and profiles of target applications. Some ontologies are already available, and have some popularity, to describe several aspects of users. CARFO ontology does not aim to replace these ontologies but to reuse them.

FOAF Ontology

The Friend-Of-A-Friend ontology[8] was one of the first popular vocabularies in the web of data (Brickley, 2010). It is a lightweight RDF(S) vocabulary. It provides a set of concept and properties to describe people and social connections. The main concept is obviously foaf:Person. Being an RDF resource, a foaf:Person is identified by means of a URI and it is characterized by attributes, such as name (foaf:name), mail (foaf:mbox), birthday (foaf:birthday), etc. The identity of a person in the web can also be described by her homepage or her accounts in the social websites (for instance, LinkedIn and Twitter).

On the other hand, FOAF also enables to capture relationships between people. It provides a generic framework based on an abstract property called foaf:knows. It is assumed that any social relationship between two persons is a specialization of this property, for instance, the connections between relatives, workmates and friends.

Furthermore, in combination with web certificates, FOAF can be used to build a portable web identity, called WebID. This is a decentralized architecture of identity management empowering users to identify themselves by URIs and to control the information they want to share with third-party websites and other users.

SIOC Ontology

The goal of the SIOC initiative (Semantically-Interlinked Online Communities) is to enable the integration of online community information (Breslin, 2005). SIOC is based on an OWL ontology for representing rich data from the social web (blogs, social networks, mailing lists, etc.) in RDF. In the past four years, SIOC has recently achieved significant adoption through its usage in a variety of commercial and open-source software applications. SIOC is designed to be fully compatible with the FOAF vocabulary for expressing personal profile and social networking information.

Current online-community sites are isolated from one another. The potential synergies among many sites, communities, and services are expensive to exploit, and their data are difficult and cumbersome to link and reuse. For a instance, parts of the answer a person is looking for are implicit in various discussions of a number of online communities, but people participating in one discussion can't readily access information about related discussions elsewhere. The main reason for this lack of interoperation is that for the most apps in the social web (from Facebook and Twitter to private chats), common standards still don't exist for knowledge and information exchange and interoperation. SIOC's ultimate goal is to fill this niche providing an RDF-based format for social data exchanging.

The central concept of the ontology is sioc:UserAccount, which captures a given user in an online community site. A foaf:Person is normally registered on a site through a user account. The property foaf:holdsAccount enables cross-links between people and their multiple accounts. Notice that the property sioc:account allows establishing reverse relations between a sioc:UserAccount and the foaf:Person. SIOC also introduces a set of properties and concepts to provide descriptions about the user generated content, such as posts in a weblog and messages in a chat board, represented as instances of sioc:Post. Posts can also be threaded whether they are connected by a common subject or by reply. Furthermore, SIOC data model also covers the channels

[8] http://foaf-project.org/

where these discussions are made (sioc:Forum), which can be linked to the web sites that host them (sioc:Site).

SIOC enhances the description of both forums and posts by means of SKOS concepts in order to create mediated links between user-generated content. These relations enable the navigation between several kinds of resources, meeting the “linked data” initiative.

WAI Ontology

The Who Am I![9] (WAI) vocabulary aims to extend FOAF through introducing the concepts of roles and profiles. In the real world, people are more than just persons, they might be musicians, presidents of government, firemen, football players or car drivers in a traffic jam. Moreover, people modulate their personality to the pertinent situation or context. For instance, John as a member of the last.fm community expresses some musical interests, which can be used to find like-minded people and to recommend some contents (artists, genres, albums, etc.). John's preferences watching TV may be completely different, and it is necessary to capture this complexity inherent to individuals and their involvement in society.

WAI is an OWL2 ontology, designed to be fully compatible with FOAF and SIOC vocabularies. It provides a flexible mechanism for FOAF documents extension, intended to model people, specifying temporal and social features like their jobs, position in a company, tastes, security credentials or status in a given community. This mechanism is built upon two central concepts. On the one hand, WAI introduces the wai:Role class. A role is defined through a property that can be predicated of a person. In this ontology, roles are reified as first order individuals and relations between roles and players are expressed by means of the wai:plays property. WAI does not impose any *a priori* subclassification of roles. The concept is open to be refined according to domain or application requirements. However, as roles are considered instances, WAI comes up with the property wai:specializes, which enables the construction of role hierarchies, such as “student of philosophy is a sub-role of university student”.

On the other hand, WAI also introduces the concept of wai:Profile, where profiles are entities capturing the dynamic and temporal aspects of roles. The full meaning of a sentence such as “John was the sales department manager of big company from 2000 to 2007” cannot be represented by a simple relation between John and the role “sales department manager”. Profiles are introduced to cover this knowledge representation gap. Roles are not inherent to people, as they are not essential properties. A wai:Profile is a mechanism that allows referring to people when they are actually playing a given role, i.e. "person-as-role". As it occurs with sioc:UserAccount, profiles can introduce a multiplication of identities for the user as well as an increase of resources to describe a particular individual. Nevertheless, all the profiles gather together at the foaf:Person. The multiplication of identities is only apparent. Moreover, profiles are resources identified by URIs. From this perspective, profiles and roles ontological distinctions contribute to data description enrichment according to the linked data paradigm.

Profiles are also useful to represent users in different contexts, introducing them as a mechanism to provide conceptual coordinates for "contextualizing" both roles and profiles. However, no more assumptions are made about its interpretation. A natural extension of wai:Context in CARFO is made by means of temporal and geographical locations. This is a typical scenario for personalized applications and interfaces, for instance in the field of ambient intelligence or mobility, where the adequacy is made considering several relevant aspects of the user.

In addition, social communities and on-line services can be also considered as contexts for profiles. On the one hand, communities are connected groups of people which are usually materialized in the Web as generic social networks like Facebook or LinkedIn, but also as dedicated on-line communities arisen from specific web sites: last.fm for music contents and FilmAffinity[10] for movies are good examples for this case. In

[9] http://purl.org/wai
[10] http://www.filmaffinity.com/

conjunction with FOAF and SIOC, social communities representation could benefit from WAI profiles management. When contexts and communities are used to fix the interpretation coordinates of the profile, roles may be implicit. In this case, a profile is considered a "person-at-context" or a "person-in-community", rather than "person-as-role".

FRAP Ontology

Preferences are an important part of user profiles for many applications and user-oriented tasks. Preferences are statements of the form “Alice prefers A than B” or “Alice thinks A is better than B”. Basically, preferences are user modal attitudes about objects and situations of the world. Despite a considerable number of proposed languages for representing user preferences, effective publication of this information in an exchangeable format is far from being a reality.

One of the most successful proposals is the Review vocabulary[11], a lightweight OWL ontology intended to capture rating and reviews. However, this vocabulary lacks the expressiveness of complex preferences, such as “the user prefers terminals with high screen resolution” and “Alice likes Metallica albums, but not Jon Bon Jovi ones”. Other efforts have been made to represent user preferences in particular domains. The CC/PP vocabulary[12] is a W3C initiative expressing device capabilities and user preferences to guide the adaptation of delivered content (Klyne, 2004). However, CC/PP preferences are limited to desired attributes of device components. In addition, it is necessary to eliminate expressivity restrictions in CC/PP, which do not exist in RDF. CC/PP defines a hierarchical structure based on two main levels (components and attributes). It means a significant restriction over RDF, as its expressiveness is considerably reduced. In practical terms, CC/PP could be seen as a kind of big table in the form key-value, where content providers are very restricted in what regards to the semantic relationships they can define. In the multimedia domain, an OWL ontology has been proposed to specify how to combine content filtering and browsing criteria with Boolean operators. These kinds of preferences are designed to be applied to MPEG7 and MPEG21 specifications (Tsinaraki & Christodoulakis, 2005).

A closely related initiative is (Kießling, 2002), where a language for preferences for querying databases is defined. This approach introduces an algebra and operators to capture “wishes” of users. This formal, abstract language is then translated to extensions of SQL and XPath for relaxed queries. As it is not guaranteed that there will exist exact matches for all the conditions of a given query, preferences allow looking for the best possible matchmaking.

The Framework for Ratings and Preferences Ontology[13] (FRAP) proposes language for representing and exchanging preferences as RDF data in the web. Moreover FRAP is domain-independent, and it is designed to be compatible with any other existing RDF(S) and OWL vocabulary, as well as domain objects described as linked data (for instance, DBpedia resources). The FRAP ontology distinguishes between two complementary notions of preferences:

1. *Preferences-as-constraints*, i.e., conditions about preferred attributes of the resources. A constraint is a set of qualitative descriptions of the desired attributes that objects must ideally satisfy in order to be of interest for a user.
2. *Preferences-as-ratings*, i.e., quantitative measurement of the “appealingness” of a particular object to a user. A rating captures the user satisfaction with respect to a given object within a scale of numerical values (also called utility). Recommendation systems typically use the real interval [-1,1] for calculating final utilities, while application front-ends measure users’ opinions with discrete scales, like the five stars classification used by Amazon or YouTube.

This lightweight vocabulary provides domain-independent means to describe user profiles in a coherent and context-aware way. FRAP has been designed as an extension of both Friend-Of-A-Friend (FOAF) and Who Am I! (WAI) ontologies.

[11] http://vocab.org/review/terms.html
[12] http://www.w3.org/TR/CCPP-struct-vocab/
[13] http://purl.org/frap

FRAP introduces the concept frap:Preference which reifies the relation between a user profile and a constraint. This relation is realized by means of the property frap:holds. Moreover, the auxiliary concept frap:Pattern is provided in order to capture the constraints comprising a given preference. For instance, a preference such as “Alice likes smartphones with Android operative system”, shown in Figure 2 - FRAP representation of "Alice like crime films starred by Spanish actors", is built upon two complementary constraints. The first constraint is crime films. The second one refers to actors born in Spain. The concept frap:Pattern is the mechanism that enables reusing other domain ontology, in this case the DBpedia vocabulary, providing integrity to the complete preference expression. This mechanism allows compositionally building complex constraints in RDF.

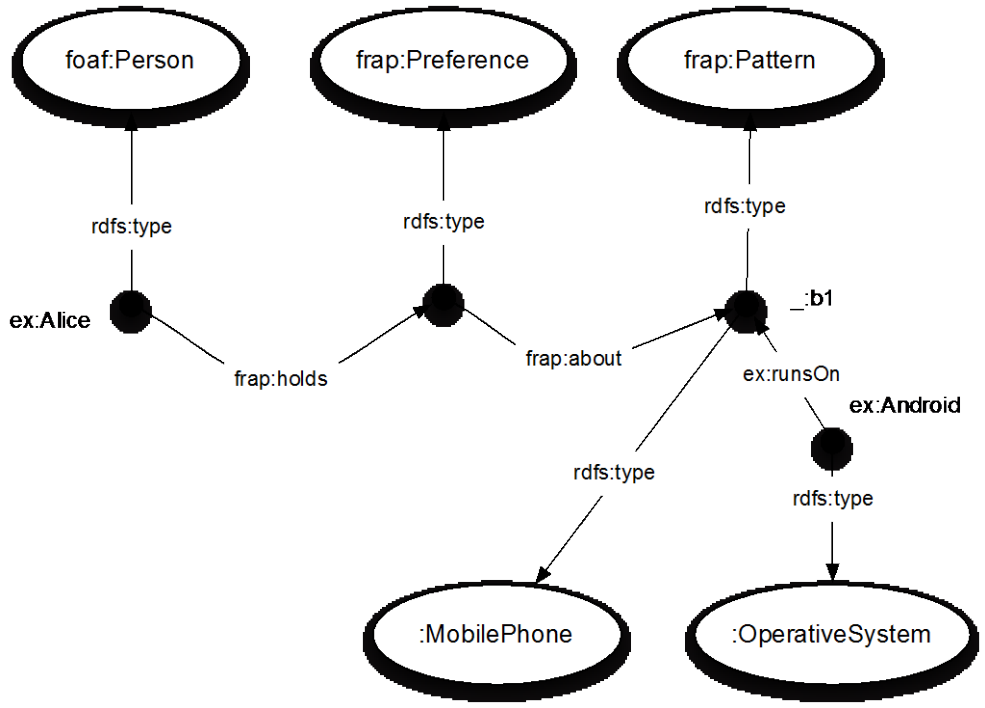

Figure 2 - FRAP representation of "Alice like crime films starred by Spanish actors"

Regarding ratings, a ternary relation between a user, a item (any RDF resource, including instances of frap:Preference) and a utility value, the ontology introduces another concept, frap:Rating, which reifies this tuple. Three specific properties capture the relationships between the rating and the user (frap:assignedBy), the item (frap:rates) and the utility value (frap:utility).

FRAP plays an important role in CARFO design as it enables the representation and exchange of user preferences across applications and scenarios. CARFO may also benefit from FRAP preferences as they can be translated to rule and query languages, thus these preferences can be of interest of, on the one hand, recommendation systems that may filter content to be presented in final user interfaces. On the other hand, SERENOA runtime might consider these preferences for the interface generation and adaptation process, using these preferences to accommodate applications front-ends to specific need and profiles of users. Furthermore, being FRAP a domain-independent vocabulary opens the door not only for a reutilization of preferences themselves, but for reusing existing domain ontologies that actually serve as data models for some given applications and services.

This section has presented four fully compatible OWL vocabularies to be reused in the context of the CARFO ontology for user representation. Reusing existing data models embraces the linked data and semantic web initiatives and facilitates data exchange and interoperability among applications and servers. This way, SERENOA builds on top of widely adopted vocabularies that can be considered as *de facto* standards, and widens its scope outside the limits of the project and the consortium members.

As an overview, Figure 3 - Combination of user-oriented ontologies in CARFO illustrates the compatibility of the above-presented vocabularies. Notice that an application can combine and extend concepts and properties, adapting them to its specific requirements.

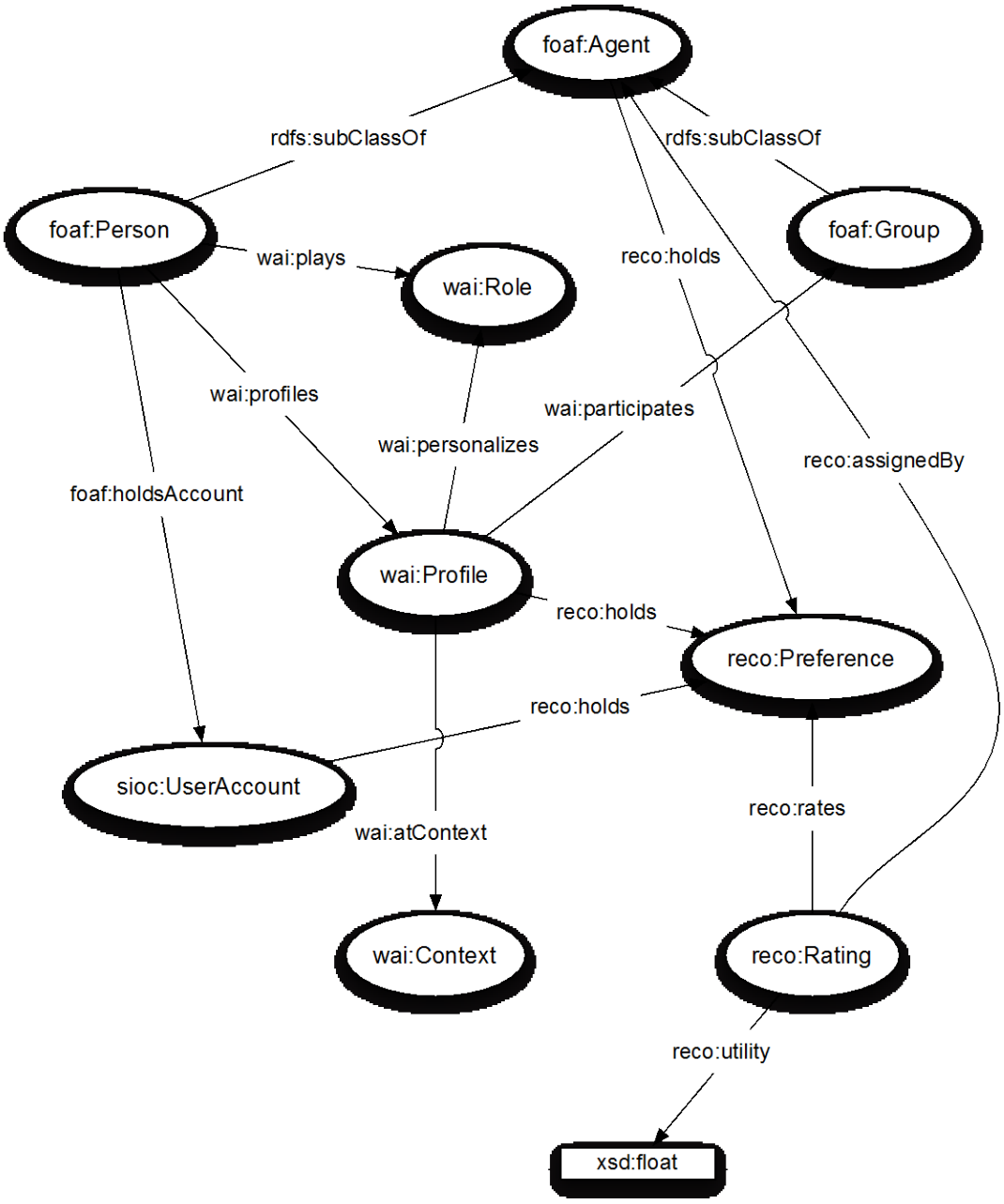

Figure 3 - Combination of user-oriented ontologies in CARFO

2.3.2 Platform

In the CARFO ontology, the term “platform” comprises all the elements of hardware, software and networks relevant to describe a computing environment. This is a much broader concept than just a server or client side framework, such as MyMobileWeb or Android. Therefore a comprehensive description of the platform is challenging, because it involves different abstraction levels, a wide range of devices and complex software applications, such as web browsers and operating systems.

This deliverable puts the focus mostly on device description as a starting point, exploring the design space and state of the art solutions.

UAProf (Forum, 2001) is a vocabulary proposed by the Open Mobile Alliance from the CC/PP specification defined by the W3C through the extinct Device Independence working group, which is expressed in RDF. UAProf profiles are created as documents expressed in the homonymous vocabulary. They are referenced by means of a URI provided by some web browsers (generally, a significant amount of mobile web browsers) in their HTTP requests. Both UAProf and CC/PP offer an interesting framework for device description (other well-known databases include WURFL, DeviceAtlas and Alembic). They have been providing device descriptions used by the software industry over the last decade. Hundreds of device models expose their software and hardware characteristics by means of a UAProf document which may be cached and then enhanced.

It is important to note that UAProf documents contain static device descriptions. This means that they contain information that is known *a priori*, such as screen resolution, Bluetooth profiles supported or the web browser(s) installed from factory. Some examples of dynamic device information are battery charge level, or screen orientation (landscape, portrait).

Previous research work has considered UAProf and CC/PP limitations. The first efforts in the analysis of these specifications (Butler M. H., 2002) (Butler M. H., 2002) reflect the absence of a formal specification for profile resolution, the lack of a mechanism to allow combining profiles expressed in different vocabularies and the need for a formal definition of vocabularies –unfortunately, often indicated as comments in UAProf profiles.

In (Indulska, 2003), a CC/PP-based vocabulary is proposed in order to represent more detailed context information for content and software adaptation. One of the most relevant problems found in CC/PP is the organization of device description in two layers. This forces the use of undesired syntactic sugar to express some definitions and relationships between device properties.

Related to the aforementioned work, an interesting study of the limitations in CC/PP and UAProf is presented in (Gergič, 2008). Its conclusions state that basing CC/PP on RDF does not seem very appropriate as it basically models a hash table with name-value pairs. The study also considers that CC/PP and UAProf describe the data structures in which device profiles are represented but they do not provide an API to access the properties contained.

Sometimes, information about a generic software component (for instance, web browser) is provided in a UAProf description. Actually, a device may include more than one implementation of that software component (for instance, two or more web browsers) without its UAProf clarifying whether both of them are compliant to the description. UAProf lacks the ability to express these implementation details.

CARFO device descriptions are built on the experience of UAProf, and seek backwards compatibility in order to make use of the extensive existing descriptions available in UAProf repositories. The remaining of this section addresses three-design issues related device description. We describe generic solution patterns that do not just apply to device description, but to all the platform components.

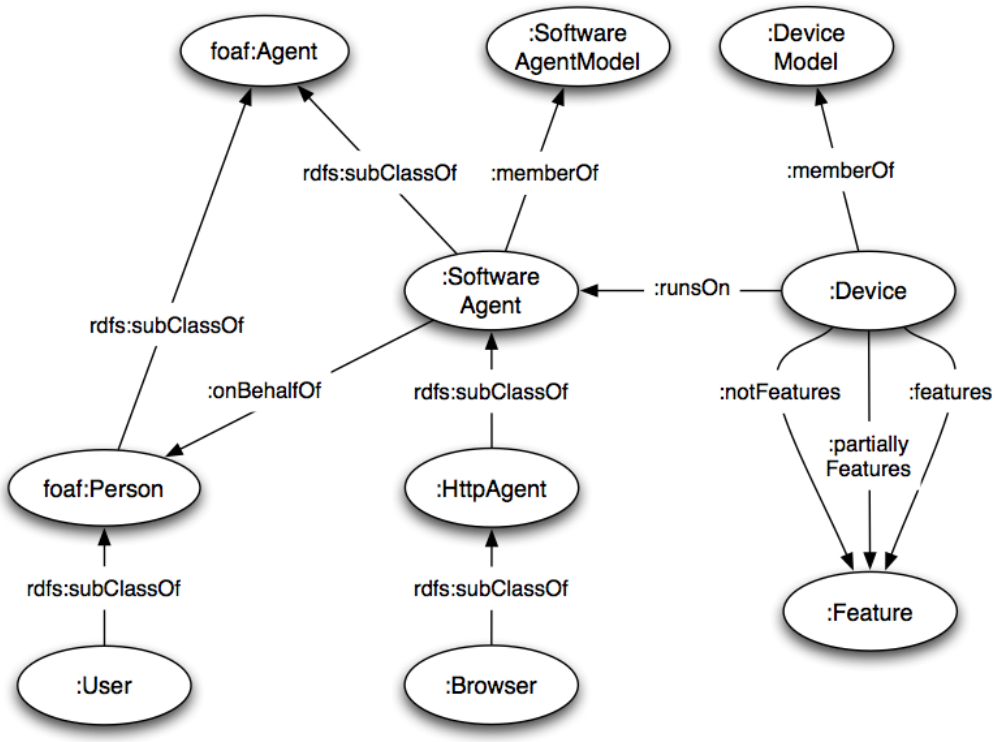

Figure 4 - Upper level classes of the platform Vocabulary

2.3.2.1 Hardware and software

A device (hardware) is an aggregation of components, based on a blueprint. For instance, a mobile phone comprises a display, a keyword, antenna, battery, CPU, memory, etc. Moreover, these components can usually be disaggregated into smaller parts, i.e., they are themselves artefacts. CARFO introduces the concept :Device to capture hardware elements.

The CARFO ontology introduces two properties to capture the structure of any artefact, including devices: hasPart (and its inverse isPartOf) and component (and its inverse componentOf). Both properties have similar semantics, enabling to capture the mereological[14] relationship between a given artefact and its constituents. The former is a transitive property, which is useful for some purposes but as a consequence the order relation is lost in the case of complex hierarchy. On the other hand, the latter is suitable to keep separated the different levels of the hierarchy. Notice that both properties are anti-symmetric. In other words, two distinct entities cannot each be a part of the other.

This solution is inspired by the upper-level ontology DOLCE, which identifies a set of properties to represent different mereological relationships (such as temporal and spatial inclusion). Other ontologies, like Dublin Core[15], also provide machinery to capture the relation between a whole and its parts, in this case, applied to information resources (multimedia documents).

In the case of software, such as browsers, social networks, desktop applications and so forth, CARFO introduces the generic concept :SoftwareAgent. Each type of software program is considered as a subclass of this top-level one: e.g., :Browser. The relationship between software agents and hardware devices is captured by the :runsOn property.

Figure 4 - Upper level classes of the platform Vocabulary presents the top-level architecture of CARFO's understanding of the platform. Following sections detail modelling decision about specific aspects of this

[14] "Mereology" is the "theory of parthood relations: of the relations of part to whole and the relations of part to part within a whole" as defined in the Stanford Encyclopedia of Philosophy
[15] http://dublincore.org/documents/dcmi-terms/

conceptual design.

2.3.2.2 Products and product-types

One of the modelling challenges of the CARFO ontology (and artefacts in general) is the distinction between concrete products and product-types (also known as models). The usage of a single class, for instance to capture devices, leads to some confusion and inconsistencies as detailed below:

1. **Indiscernible resources**. It is not possible to semantically distinguish between one instance representing a device model and another representing a particular device because both individuals inherit the same properties from the class they belong to (i.e. Device).
2. **Incoherent updating policy**. Particular devices are subject to changes: (a) due to the dynamical evolution of the context, such as the battery level, the signal power and environmental parameters (location, temperature, etc.); and (b) users might personalize properties of their mobile phones. A mobile phone might be connected with other devices within its local environment, such as an external display, a print or just another mobile device. On the other hand, models are invariant descriptions of types of products, where updating rules only apply when an evolution of the product is placed on the market.
3. **Inconsistent descriptions**. There is not a mechanism to distinguish between models and particulars. Therefore, automatically checking the semantic consistency of the instances of Device is far from being a trivial task. For instance, consider a device model (such as the HTC Touch Diamond). If there is a contextual property (e.g. location) applied to this product-type, it is not possible to detect *a priori* that there is an inconsistency.
4. **Redundant information**. Apart from modelling issues regarding products and product-types, inheritance mechanisms are not clearly defined. CC/PP approach proposes a vocabulary where properties of device models are treated as default features, and the user (or developer) can overwrite these values. However, this implies that default properties of a device (i.e. factory defaults) have to be repeated for every terminal of the same model.

CARFO proposes a new concept Model to capture these collections of products. In this context a 'model' represents a type of a product, where product is understood as an artefact, i.e., anything that has been industrially created by a company and offered to the market. This way CARFO is able to distinguish between concrete devices, such as the mobile phone, the laptop and the camera of John Phillips, and models of these products, namely an iPhone 3GS, a Sony Vaio and an Easyshare M590 Kodak Camera.

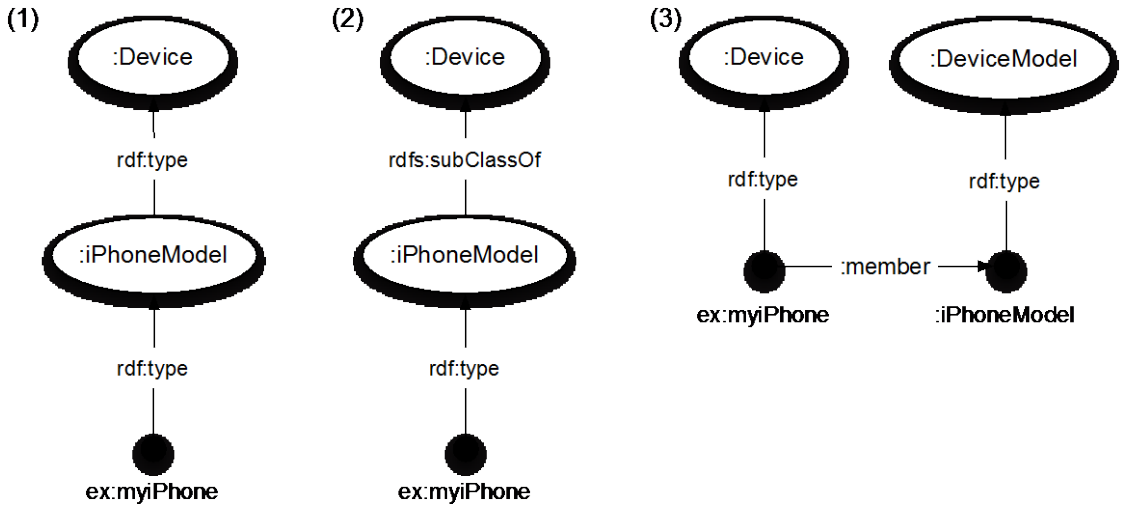

Figure 5 - Proposal for modelling product-types

In D2.2.1 CARFO (R1), three distinct alternatives to formally capture the concept Model in an OWL ontology were explored, using the running context of device modelling. Figure 5 - Proposal for modelling product-types graphically presents these alternatives deemed:

1. “Metamodelling” alternative. In this case, product-types are considered both as instances and classes, what is possible to be represented in OWL2.
2. “Product-types as classes” alternative. This approach interprets product-types (models) as ontology classes. Notice that the difference with the above technique is that no metamodelling is introduced, i.e., product-types concepts are subclasses of models.
3. “Product-types as instances” alternative. This approach considers product-types as instances. This means to reify product-types as collections of products. This way, product-types are instances of the class :Model (or :Device in the context of SERENOA), while products (e.g.: particular devices) are instances of :Product. To retain the relationship between a product and its product-type, the property :memberOf is introduced.

We have finally chosen the last one as it provides some key benefits. This design pattern firstly introduces product-types as part of the domain, without including metamodelling. It is possible to describe a model as any other resource. Secondly, the design pattern provides RDF-queriables descriptions of product-types, which can be performed by means of the SPARQL standard query language. The main disadvantage of this approach is however that the instantiation relation between a product-type and its products is lost. The property memberOf does not have a special status in the ontology. Therefore, no inheritance mechanism can be used between the two entities.

Last but not least, this approach enables the reconciliation of factory defaults and customization features of a given product. Figure 6 - Example of a device description using the "product-type as instance" approach shows how a device personalization is combined with the description of its product-type using this design pattern. The descriptions of the iPhone4 and Nexus1 models cover their complete technical specifications: number of megapixels of the camera, operating system, battery, connectivity, etc. This means that a particular device should match this specification in order to be considered a member of this model, i.e, to be catalogued as an iPhone4 or a Nexus1. Figure 6 - Example of a device description using the "product-type as instance" approach represents the display width of two devices (ex:device1 and ex:device2) that belong to different product-types. This information is needed to correctly adapt the delivery content to the phone, but the width value for the iPhone device is unknown. The goal is to retrieve this information from the RDF graph asking the value, in one case, directly to the terminal properties (the Nexus1), and in the other case, indirectly through its product-type specification (the iPhone4).

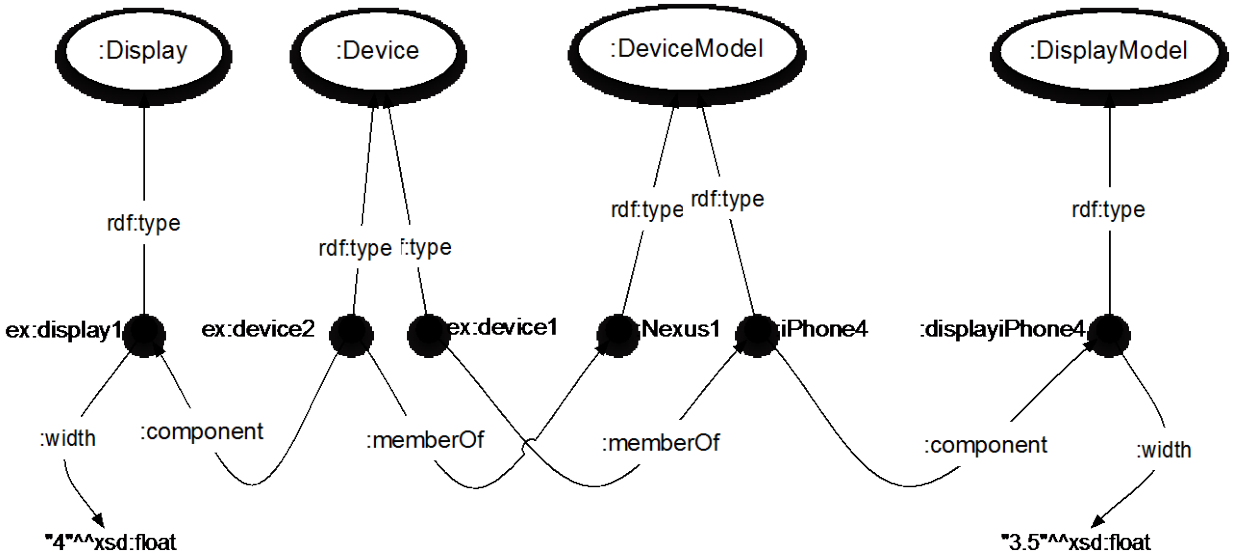

Figure 6 - Example of a device description using the "product-type as instance" approach

We use SPARQL language to query the RDF dataset. The following SPARQL query demonstrates the union of two graph patterns. The query firstly tries to find whether there exists a known value of the terminal that matches the first graph pattern (i.e. the width of the display). If there isn't any known value for this property, the query will check whether the technical specification of the device's model does match the second graph pattern of the WHERE clause. This second pattern of the union expression simulates Negation as Failure behaviour, obtained by a complex graph pattern that combines an OPTIONAL and !bound FILTER expressions.

```
PREFIX ex: <http://www.example.org#>
PREFIX serenoa: <http://www.w3.org/2001/di/Group/Ontologies/DeliveryContext.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?device ?width ?model
WHERE {
{ ?device serenoa:component ?display .
  ?display rdf:type serenoa:Display .
  ?display serenoa:width ?width . }
UNION
{ ?device serenoa:memberOf ?model .
  ?model rdf:type serenoa:DeviceModel .
  ?model serenoa:component ?displaymodel .
  ?displaymodel rdf:type serenoa:DisplayModel .
  ?displaymodel serenoa:width ?width .
OPTIONAL {  ?device serenoa:component ?display .
```

The execution of the SELECT query returns the results presented in Table 1 - Results of the product-type factory defaults query. Notice that the ?model variable is bound only for those rows in which the default value is returned. This way, it is possible to distinguish between the default and concrete values returned:

?device	?width	?model
ex:device1	“3.5”	:iPhone4
ex:device2	4	

Table 1 - Results of the product-type factory defaults query

Another relevant aspect of this design pattern of product-types is how sub-classification is accomplished. Consider for example the car model: “Volkswagen Golf”, which is a car manufactured by Volkswagen since 1974 and marketed worldwide across six generations (there have been many configurations of this car till

nowadays). Each of these configurations is a new product-type that is related with the general definition of the "Volkswagen Golf" model but introduces some modifications and specializations in the description of the model (engine, transmission, wheelbase, number of doors, etc.).

The hierarchy between product-types is easily captured when they are treated as classes, but not when they are individuals. Inheritance is not defined for domain data, i.e., the subclass relationship is not applicable to first-order entities. However, the specialization of product-types is relevant for CARFO because there are groups of device product-types that internally sub-specialize themselves. For example, for the iPhone 3G device model description, all device instances share the display size, the audio support, power and battery, etc. However, there are two available versions of the phone, one with 8GB flash memory, and another with 16GB. To this end, a new property specializes is included in the ontology. This property enables relations between product-types, i.e., establishing a partial order or hierarchy: if a product-type A specializes a product-type B, then product-type A shares the properties of B, but B doesn't share properties of product-type A.

2.3.2.3 Capabilities

Hardware and software components from the same vendor or the same family usually share common features. It is just natural to organize the information to minimize redundancy by creating descriptions that refine or extend other descriptions. The representation of these relations and their semantics are not straightforward in RDF.

In some cases, the different values that an attribute in UAProf may take are strictly defined. This leads to incoherent device descriptions in the sense that, for instance, an attribute is valued with a string for which there is no formally specified format. As an example, consider the values for the attribute *BluetoothProfile* in a set of actual UAProf description files downloaded from different manufacturers' repositories. A quick read after the values accepted show that the support for the AVRCP Bluetooth profile is noted with different strings (in alphabetical order): *"audio video remote control"*, *"Audio Video Remote Control Profile"*, *"Audio Video Remote Control"*, *"Audio Video Remote Control – Target"*, *"Audio Video Remote Control Profile"*, *"audiovideoremotecontrol"*, *"AVRCP"*, etc.

Some other typical inconsistencies include the expression of the values of a same attribute by means of different types. Following the same procedure for the *BluetoothProfile* attribute, a study of the *NumberOfSoftKeys* attribute has been carried out. In addition to the expected *xsd:integer* values (0, 1, 2, 3, 4, etc.), a "None" value has been found for many Motorola devices, such as the A1600. This is due to different versions of the CC/PP schema, UAProf vocabulary and third-party schemas (such as those from the 3GPP) published over time.

Ontology languages on top of RDF, such as RDF Schema and OWL, bring in the ability to declare certain constraints. However, some of the aforementioned integrity constraints are beyond the expressiveness of these languages. Even those potentially usable lead very often to consequences that are not intuitive for people trained in databases and XML. Thus, it may make sense to introduce an *ad hoc* validation tool that implements the logic behind semantic restrictions. Finally, UAProf documents offer many chances for improvements in the light of recent developments in linked data (Heath, 2011). More specifically, UAProf documents, profiles and the resources they contain should be assigned HTTP-resolvable URIs. By doing so, they become extensible and linkable, and new opportunities appear for re-using shared resources.

Capabilities are captured in CARFO by introducing a new upper-level concept: :Feature; and a set of upper-level properties to describe hardware and software components. These properties are used later on during the ontology population process:

- :features, which indicates that a component (or model component) supports a given capability.
- :notFeatures, which indicates that a component (or model component) does not support a given capability.
- :optionallyFeatures, which indicates that a component (or model component) might support a given capability.
- :hasPolyfill, which indicates that a component (or model component) does not support a given

capability, but has polyfill.

2.3.2.4 CARFO platform ontology

From the analysis of the above sections, the CARFO platform module (i.e., main classes and properties) can be organized according to a 4-axes representation (see Figure 7):

1. Hardware (i.e., physical components and devices)
2. Software (i.e., software programs and agents)
3. Product-types (i.e., invariant descriptions of hardware devices and software agents models).
4. Particular products (i.e., actual devices and software programs owned by a given user).

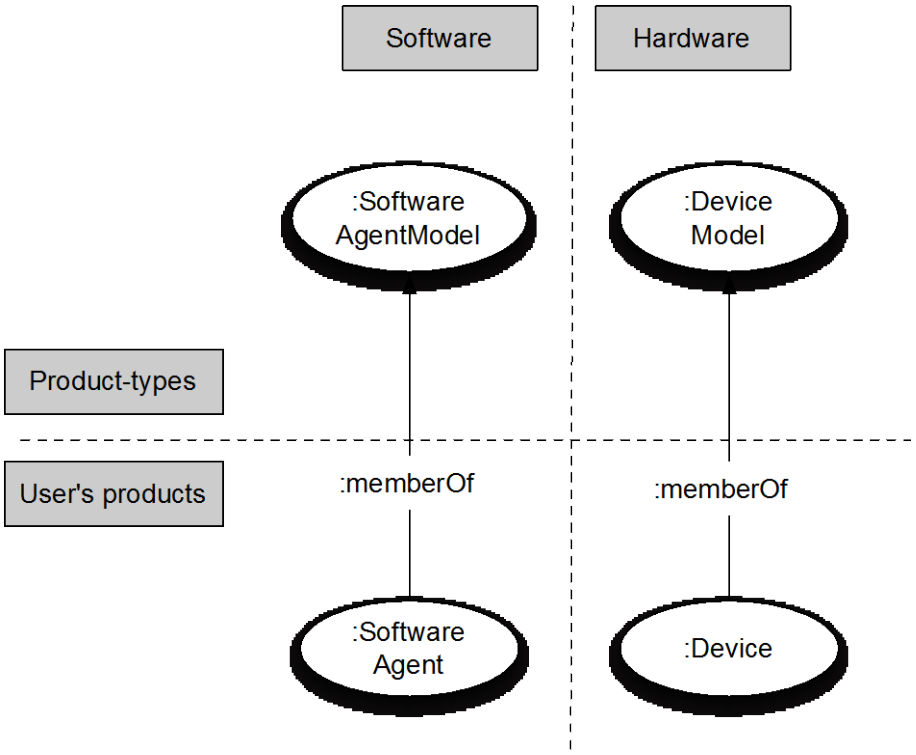

Figure 7 - Representational axes of the CARFO platform module

2.3.3 Environment

The identification of information coming from the context of use during the interaction with systems enables several adaptation techniques to be performed, such as: personalization, multi-delivery and location-based services. The customization considers the context, e.g., users' preferences, device characteristics, or bandwidth restrictions, allowing applications to be adapted accordingly. The context may impact which contents are presented, how and also how the navigation is organized with respect to both semantic and syntactic properties, as such context information should be taken into account in all phases of the development process (Schwinger, 2006).

The third pillar of the Context of Use module is the environment surrounding the user and the platform. In CARFO, the environment is interpreted in a broad sense, comprising not only ambient conditions (such as temperature, geographical location, time, etc.) but also contextual ones (such as "I'm at home" or "I'm busy at work"). The "environment" is defined then as the information or knowledge that surrounds an entity, physical or conceptual. The environment of an entity is, strictly speaking, infinite. However, the relevant context of an entity is finite and enclosed.

2.3.3.1 Computational patterns

There are four big methods of using the environment and contextual information. Such four methods are:

1. The environment as an additional input of data. This pattern takes into account the environment as a parallel input to the information brought in by the user

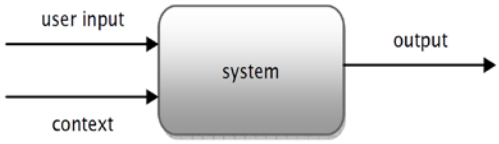

Figure 8 - The environment as an additional input of data

2. The environment as a modifier of the input. In this pattern the environment is used to modify the input of data of the user, making easy to identify the objectives of the user.

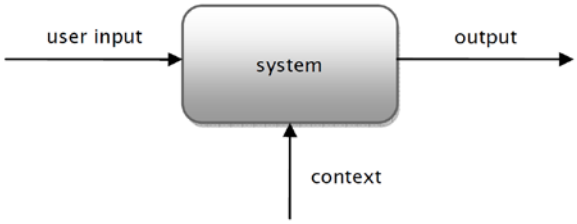

Figure 9 - The environment as a modifier of the input

3. The environment as information returned to the user. This pattern represents a closed loop, in which the environment offered by the system could modify the inputs of the user, generating new outputs.

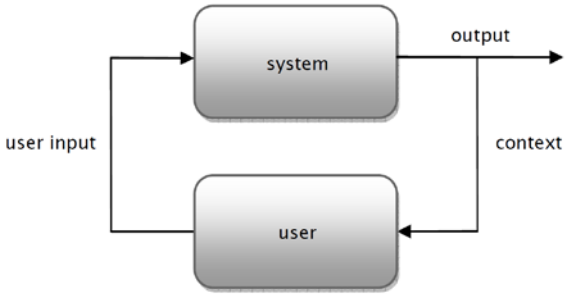

Figure 10 - The environment as information returned to the user

4. The environment as a trigger of events. This pattern uses the contextual information as a primary input. That contextual information, used among some activation rules provides the system a way to fire a new event in the system.

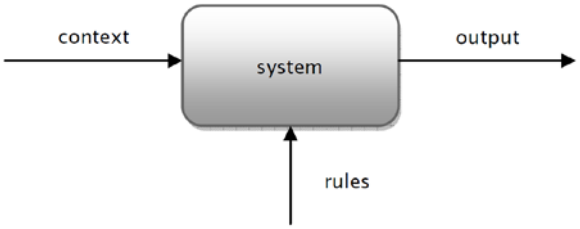

Figure 11 - The environment as a trigger of events

2.3.3.2 Types of environment

There are several types of environment depending of which characteristics we focus on:

- Primary or secondary: the environment can be classified as a hierarchy, where the information in upper levels encapsulates data in lower levels.
- Internal or external: the environment can be internal or external, depending of its dimension with respect of the user.
- Physical or logical: if the information of the environment can be measured with sensors (such as location, light, temperature, etc.…) we are dealing with a physical environment. However, if the information is obtained monitoring user tasks, emotional conditions, tools used, etc., we are talking of a logical environment.
- Low or high level: low-level information of the environment is that information sensed and stored without the need of any type of processing or combination. Values measured from the sources of the environment come directly in the proper way. However, high-level information is that information which is output of the processing of the low level information. We are able to obtain conclusions (new environment) as a result of this processing.

2.3.3.3 Characteristics of the environment

The environment has several characteristics that should be taken into account:

- Temporary. The information of the environment can be static or dynamic. Static information is constant, but dynamic information changes over the time.
- Accuracy. The information of the environment is accurate if it reflects the reality with an error probability below a fixed and concrete value. Non-accurate hardware or imperfect logical models could arise non-accurate measurements.
- Multiplicity. Contextual information can have a unique representation used every time and everywhere (i.e. national ID of a person), or can have different facets (i.e. coordinates of the location of a person).
- Interrelation. Environment information can be related to each other. Using those relations we can derive other contextual data or higher-level environment conclusions.
- Relevance. Some kind of environment information can be more relevant than others. Therefore, we can select the suitable or more important information to the application.

Figure 12 - An overview of environmental information represented by CARF captures the main important aspects of the environment in the SERENOA project according to the work developed in the CARF framework.

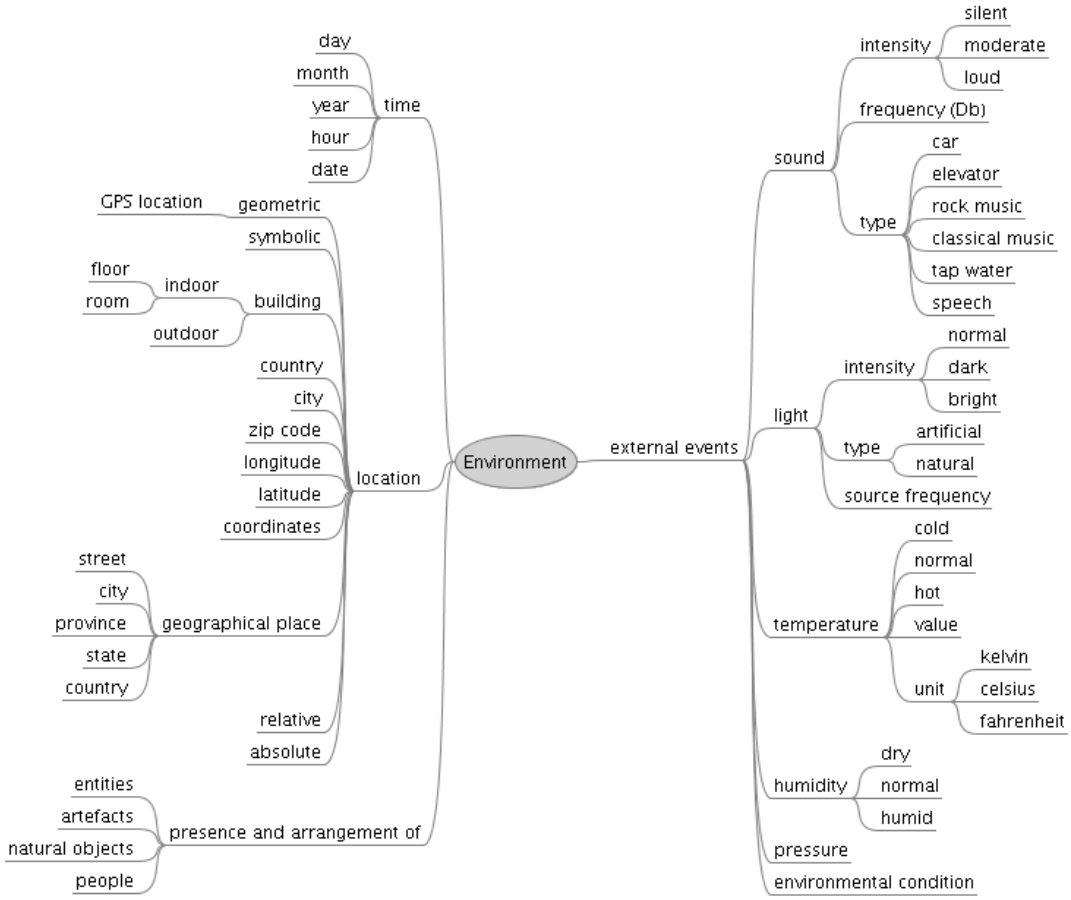

Figure 12 - An overview of environmental information represented by CARF

2.3.3.4 Description of environment

Previous efforts to describe environment include CONON, for Context Ontology (Wang, Gu, & Pung), which defines general concepts such as location, activity, person or computational entity. These terms are thought to be extensible in a hierarchical way by adding domain specific concepts. The authors divide their context model into an upper ontology and a specific ontology. On the one hand, the upper ontology is a high-level ontology that captures general features of basic contextual entities. On the other hand, the specific ontology defines the details of the general concepts and their features in each sub-domain covered.

Environment describes the situation and the environment in which the interaction takes (Brossard, Abed, & Kolski, 2011). More precisely, according to (Zimmermann, Lorenz, & Oppermann, 2007) the context regarding the environment is relevant due to the mobility of computing devices, applications and people, which leads to highly dynamic computing environments. Unlike desktop applications, which rely on a carefully configured and largely static set of resources, ubiquitous computing applications are subject to changes in available resources such as network connectivity and input and output devices. Moreover, they are frequently required to cooperate spontaneously and opportunistically with previously unknown software services in order to accomplish tasks on behalf of users. Thus, the environment surrounding an application and its user is a major source to justify adaptation operations. For W3C, the Environment denotes the set of objects, persons and events that are peripheral to the current activity but that may have an impact on the system and/or users behavior, either now or in the future (Coutaz, 2002). An environment may encompass the entire world. In practice, the boundary is set up by domain analysts whose role is to elicit the entities that are relevant to the case at hand. Specific examples are: user's location, ambient sound, lighting or weather conditions, present networks, nearby objects, user's social networks, etc.

Other significant works are related to ambient intelligence. As an example, the Amigo ontology was

developed by the FP6 Ambient Intelligence for the Networked Home Environment project. This initiative enables the description of sensor networks and environmental profiles (amigo:EnvironmentalProfile), providing a preliminary set of measurable ambient conditions.

The CARFO ontology embraces these previous works and extends them as necessary. A special solution has been developed for the precise representation of measurable quantities (not just for the environment, but also for the description of the platform). The motivation for this work arises from the limitations detected in the state of the art, namely the lack of adequate mechanism to capture semantics of measurements as RDF literals. MUO[16] (the Measurement Units Ontology) has been selected to measure environmental conditions, such as ambient temperature, or device properties, such as screen size. Notice that different measurement units are often used to measure these physical qualities (e.g., Fahrenheit and Celsius degrees, and inches, centimeters and pixels).

Figure 13 - MUO representation of area of Spain, illustrates how to accommodate the representation of the surface area of Spain's territory, according to MUO ontology.

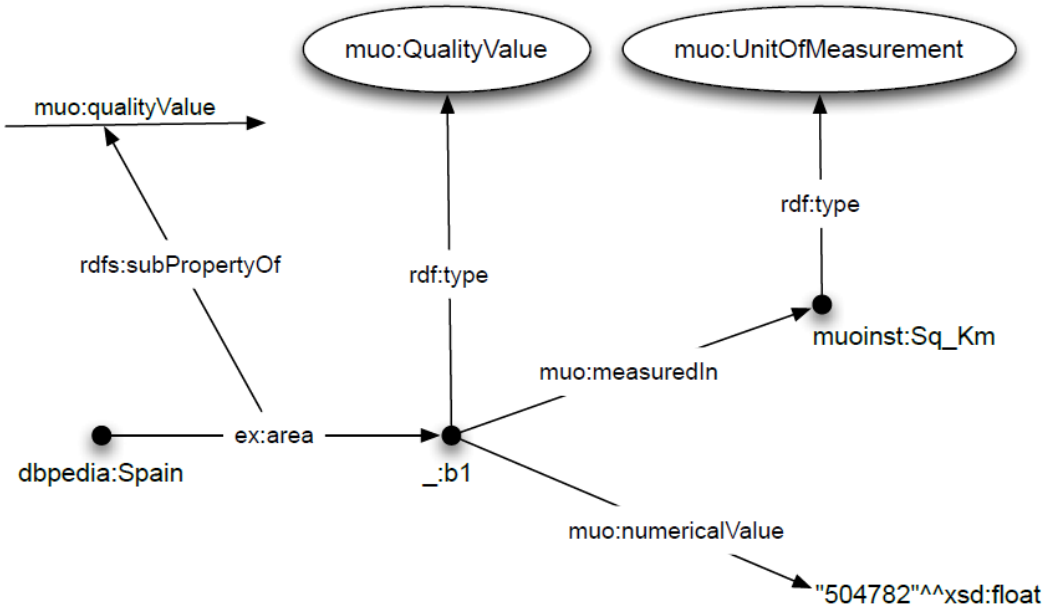

Figure 13 - MUO representation of area of Spain

2.3.3.5 Location

The notion of a location in CARFO is restricted to a set of physical locations, which include building, floor, room, street, city, country, etc. These physical locations are all assumed and have well defined spatial boundaries (e.g., all locations can be uniquely identified by geographical coordinates – longitude and latitude). In addition, all locations in a factory manufacturing area have identifiable string names that are assigned to them by some official bodies (e.g., by the factory administration).

To model physical locations, we define a class called Location, which generalizes all type of locations in a factory or building. Also, a set of properties common to all concrete physical location classes, e.g. consisting longitude and latitude, is provided.

Containment relationships were defined in order to be used between instances of Location. These relationships are defined by two related object properties called spatiallySubsumes and isSpatiallySubsumedBy. The former describes the subject of this property spatially subsumes the object of this property (e.g., a building spatially subsumes a room in the building), and the latter describes the subject of this property is spatially subsumed by the object of this property (e.g., a room in the building is spatially subsumed by the building). In the context of the OWL language, these two properties are defined as an

[16] http://idi.fundacionctic.org/muo/

inverse property of each other. In the current version of the ontology, the domain and the range of both spatiallySubsumes and isSpatiallySubsumedBy properties are of the class type Location. In other words, these two properties cannot be used to make statements about the containment of a person in a physical place.

In addition to containment relationships, physical places may also be associated with events and activities (e.g., the picker has taken all required parts from shelf 433; the picker is currently standing in front of shelf 473 etc.). To make statements about a location that is associated with some event, we introduce an object property called hasEvent, which has domain Location and range Event. Instances of Event can be associated to time intervals. We define EventHappeningNow, a subclass of Event, to represent a set of all events that are currently happening. To make statements about a location that is associated with some event that is currently happening now, we define an object property called hasEventHappeningNow.

2.3.3.5.1 Picker's Location Context example

By location context of a picker, we mean a collection of dynamic knowledge that provides geo-location (a collection of RDF statements). The location property associated to a picker is captured through the object property locatedIn. It has as range Location class and as domain owl:Thing, indicating anything (including pickers) may be located in some physical location.

Physical locations are categorized into two distinctive classes: AtomicLocation (e.g., hallways and rooms) and CompoundLocation (e.g., factory and building). Following the semantics of these two classes, we can make the following reasoning: no picker can be located in two different atomic locations at the same time, but a picker can be in two different compound locations at the same time just in case one spatially subsumes the other. This reasoning is important for detecting inconsistent knowledge about the current location of a picker.

To capture the notion, a picker can be located in an atomic and a compound location, from the locatedIn property we define two sub-properties called locatedInAtomicLocation and locatedInCompoundLocation. The former restricts its range to the AtomicLocation class, and the latter restricts its range to the CompoundLocation class. From these two properties, we define additional properties that further restrict the type of physical location a picker can be located in. For example, locatedInRoom, locatedOnFloor and locatedInHall are subproperties of locatedInAtomicLocation; locatedInFactory and locatedInBuilding are sub-properties of locatedInCompoundLocation.

3 Ontology population

3.1 What is Ontology population?

An ontology instance is an individual of the domain, such as a concrete mobile device, a given person or the company where she works. Ontology concepts are extensionally defined as sets of instances. In other words, on the one hand, ontologies explain the semantic structure of a knowledge domain by means of concepts and properties. On the other, instances and relationships between them are the domain individuals and facts that are described according to this semantic structure. An ontology population task is then the process to fill up a given ontology providing domain instances and relationships described according to its semantic model. Using the Description Logics terminology (as the CARFO ontology is described using OWL2), the population process would affect the so-called A(assertion)-Box. Notice that an ontology population does not change the semantic structure of an ontology, the T(erminology)-Box, as the concepts hierarchy and their axiomatic descriptions are not modified.

3.2 Process of Ontology Population

The approach proposed for this task, from a high-level perspective, is actually the result of a linear workflow, which is a simplification of the approach presented in (Petasis, Karkaletsis, & Paliouras, 2007).

1. The first step is, obviously, the identification of suitable information sources, relevant for each module of CARFO. These information sources could be unstructured (e.g.: corpora), semi-structured (e.g.: spread sheets) or fully structured (e.g.: databases and RDF).
2. Afterwards original data sources, where information is represented using easy-to-process formats, such as XML and JSON, are selected. These formats are prioritized to other ones as they facilitate the transformation process.
3. The original data is then programmatically transformed into an RDF dataset and load it to an inline Jena[17] Model.
4. Optionally, the data could be enriched with external sources or with knowledge derived from custom rules that are not present in the original data. This process can be done during the transformation or afterwards.
5. Finally, the generated RDF graph is serialized (preferably as RDF/XML), stored in a publicly available RDF repository and merged with the other selected and transformed sources. The web address of the endpoint is: http://data.ctic.es.

Obviously such abstract process would require to be adapted by each concrete information source. The only requirement is that all transformation processes should return the information encoded as RDF. In addition, the consortium has reached the agreement of using a common base URI for the individuals generated by all partners, which ideally should be: http://purl.org/carfo/example/.

3.3 Context of use module

3.3.1 Source identification

As the first step in the process of ontology population, three knowledge bases have been identified as relevant sources:

- caniuse.com
- CTIC Device Description Repository (DDR)
- Nokia Developer site

In the next sections, technical processes to transform these data sources into RDF are explained in detail.

[17] http://jena.apache.org

3.3.2 Automatic data extraction from caniuse.com

The website caniuse.com[18] is an online service which provides compatibility tables for the support of HTML, CSS, SVG and more features for the most popular desktop and mobile browsers. This information is potentially useful during a dynamic interface adaptation especially in the context of web applications, where web browsers are the interactive tools for end-users. At the time of this writing the original raw data is accessible online under the CC BY-NC 3.0 license[19]. This is a difference with the situation some months ago, when D2.3.1 CARFO Population (R1) was submitted. In that moment, SERENOA consortium was granted with a private access to the dataset, but the dataset was not available for all users.

The raw data is formatted using JavaScript Object Notation (JSON). Therefore a process to convert from this format to RDF is needed. After the transformation, the output RDF is comprised by two mainly information blocks:

- Browser descriptions (following CARFO model, described in Section 2.3.2.2), including feature support, as instances of the :BrowserModel concept.
- Features description (following CARFO model, described in Section 2.3.2.3), as instances of the :Feature class.

It is also worth mentioning that statistical data has been generated, using the RDF Data Cube Vocabulary[20], regarding percentage of support a feature for all the browsers, the percentage of web browser use (divided by software agent families and by version).

As a contribution to the open source and caniuse.com community, a bug was reported (and fixed) using the github bugtracker[21] related with an inconsistency in the output format of one attribute field.

As example of transformation from JSON caniuse.com to CARFO-compliant RDF, the next fragments shows the input from caniuse.com JSON format and the CARFO-compliant RDF output. Remind that these are just


```
"bb": {
            "browser": "Blackberry Browser",
            "abbr": "BB",
            "prefix": "webkit",
            "type": "mobile",
            "usage_global": {
                "10": 0,
                "7": 0.109508
            },
            "versions": [
                null,
                "7",
                "10",
                null
            ],
            "current_version": ""
        },
```


fragments, so more JSON data has been used during the input and more triples have been generated as output.

[18] http://caniuse.com/
[19] https://raw.github.com/Fyrd/caniuse/master/data.json
[20] http://www.w3.org/TR/vocab-data-cube/
[21] https://github.com/Fyrd/caniuse/issues/125

```
<qb:Observation>
    <dp:Web_browser>
      <cou:SoftwareAgentModel rdf:about="http://purl.org/carfo/example/browser/bb">
        <rdfs:label>Blackberry Browser</rdfs:label>
        <cou:engine>webkit</cou:engine>
      </cou:SoftwareAgentModel>
    </dp:Web_browser>
    <carfo:measureBrowser>0.109508</carfo:measureBrowser>
    <qb:dataSet rdf:resource="http://purl.org/carfo/carfo#dataset_caniuse_browser"/>
  </qb:Observation>

  <qb:Observation>
    <dp:Web_browser>
      <cou:SoftwareAgentModel rdf:about="http://purl.org/carfo/example/browser/bb/7">
        <rdfs:label>Blackberry Browser 7</rdfs:label>
        <nie:version>7</nie:version>
        <dct:isVersionOf rdf:resource="http://purl.org/carfo/example/browser/bb"/>
        <cou:engine>webkit</cou:engine>
      </cou:SoftwareAgentModel>
    </dp:Web_browser>
    <carfo:measureBrowser>0.109508</carfo:measureBrowser>
    <qb:dataSet
rdf:resource="http://purl.org/carfo/carfo#dataset_caniuse_browser_version"/>
  </qb:Observation>
```

3.3.3 Automatic data extraction from DDR

The Device Description Repository is a repository proposed by W3C for providing relevant capabilities of different families of devices to developers through a standard vocabulary (DDR Core Vocabulary (Rabin, Trasatti, & Hanrahan, Device Description Repository Core Vocabulary, 2008)) and an API (DDR Simple API (Rabin, Cantera Fonseca, Hanrahan, & Marin, 2008). CTIC is providing publicly and freely an online version of such service implemented in the context of the MyMobileWeb project[22], which follows the REST based Web Service paradigm. The data is based on the XML-based UAProf device profiles provided by WURFL[23] (see Section 2.3.2.3, where a detailed discussion about UAProf profiles and device capabilities is carried out). The used device catalogue is based on an earlier open source version of the WURFL dataset. This service could be used not only for populating the CARFO ontology from an initial set of devices, but also for retrieval of the capabilities of new devices.

The population from the DDR service is a two-step process:

1. Automatic extraction of the schema and alignment with the CARFO ontology. At this step, the DDR data is inspected and the implicit data model used to describe the devices is extracted. This schema is derived from the analysis of the JSON content returned by the DDR service. In order to facilitate the posterior transformation of this device repository and its integration with the other sources, the inferred schema is aligned with the upper-level schema of CARFO: :Device Model and :Feature (see Section CARFO platform ontology of this document).
2. Using this derived schema, the DDR repository is transformed to CARFO-compliant RDF graphs. As with the caniuse.com dataset, the output is loaded to an inline Jena Model and then stored in the CKB (i.e., the public RDF repository).

This data extraction and transformation process is a long-time task. More than 18,000 user agents are processed. Currently it can take more than 48 hours and we are experimenting on how to reduce this time by the parallelization of the queries.

3.3.4 Automatic data extraction from Nokia

The Nokia Developer site[24] provides machine-readable data about their products[25] according to custom ontologies developed by Nokia[26], in particular, the Forum Nokia Device Profile Ontology. This information

[22] http://idi.fundacionctic.org/DDRService_1_3/
[23] http://wurfl.sourceforge.net/
[24] http://www.developer.nokia.com/
[25] http://www.developer.nokia.com/gen/all_devices.rss
[26] http://sw.nokia.com/schemas/nokia/ForumNokia.owl

is published as an RSS 1.0 channel, using the RDF triples-based data model. This dramatically simplifies the integration of this information in the CKB. Nevertheless, data mediation is needed between the CARFO ontology and the Nokia ones. The followed strategy is to load the RDF coming from the RSS channel and to query about specific information relevant to CARFO purposes, i.e., Nokia devices description. The above Nokia ontology is aligned with the CARFO ontology, as the latter provides an upper-level perspective and is not dependent with specific modelling decisions of Nokia.

Finally, as with caniuse.com and DDR dataset, the outcome is uploaded to the CKB from an intermediate Jena Model.

3.4 Ontology population in numbers

The output of the ontology population process, in the first version and in this final version, can be briefly summarized with the following figures:

	Statements (RDF triples)	Devices models	Browser versions	Different features
CARFO R1	5.863.117	339	9.924 (264 different browser models)	1.502
CARFO R2	10.864.325	363	18.305 (311 different browser models)	1.632

Table 2 - Statistical difference between R1 and R2

3.5 Tools and Technologies

The CARFO population task has been undertaken on a fully automatic way from the different sources of information described above (see Source identification). The toolkit covers not only the aspects related with the population, but also the automatic derivation of specific parts of the ontology from the original source schema (such as the DDR). This system has been developed in Java, the Apache Jena[27] Framework as the core technology for dealing with RDF, and using SPARQL (Prud'hommeauc, 2008) as the main language and protocol to cope with the data. A wider overview of the suitable technological environment that could be used to extend or improve such experiments can be found at W3C Semantic Web Standards wiki[28].

The developed source code is publicly available from the subversion repository of the project at Morfeo Forge[29] under the open source GNU LGPL license.

[27] http://jena.apache.org/
[28] http://www.w3.org/2001/sw/wiki/Category:Tool
[29] https://svn.forge.morfeo-project.org/serenoa/trunk/carfo/api

4 Ontology access

4.1 Introduction of the CARFO Knowledge Base (CKB)

The CARFO Knowledge Base (CKB) is the main source of contextual information, well-structured and backed by the CARFO ontology, to support the advanced adaptation of SFEs. As described in the previous chapters, the CARFO ontology mainly supports the Context of Use axis of the CADS framework.

4.2 Publication of the CARFO Ontology

The publication of any ontology usually includes the reservation of an ontology namespace, i.e., the URI that identifies the ontology. It is also necessary to host the ontology files in a public repository and associate the URI with this hosting location. The official name of the ontology has been decided by the consortium to be CARFO (CARF Ontology). A namespace has been registered at PURL for the ontology: http://purl.org/carfo. The ontology is hosted by CTIC in http://vocab.ctic.es and is published according to W3C best practices (Berrueta & Phipps, 2008).

Human-readable documentation of the ontology is provided to facilitate CARFO consultation and consumption. Best practices recommend ontologies to include valuable metadata information (Tejo-Alonso, Berrueta, Polo, & Fernández, 2012) that helps to identify the elements of the ontology that can be useful for potential users. In the last years, a number of ontology documentation tools can leverage that metadata to offer visualizations of this information in the form of HTML pages or other multimedia formats to the final user. SERENOA uses CTICs Parrot[30] online available tool to automatically generate the end-user documentation of the CARFO ontology.

4.3 Publication of the CARFO Knowledge Base

All the transformed data is available online through an SPARQL endpoint: http://data.ctic.es/sparql, at the named graph http://purl.org/carfo/example.

Of course, dumps of the data, partial or full, can be requested by third-parties (outside the consortium) who would be interested on further exploitation.

4.4 Experimental Evaluation of Ontology population (Querying the CKB)

This section describes the experimental evaluation of the ontology population, where we performed a set of queries by interacting with the CARFO Knowledge Base (CKB) (more details in Section 4.1) in order to fetch different sets of data. In the following paragraphs, a subset of SPARQL queries[31] is outlined.

4.4.1 Query 1

Query 1 gets the set of versions of mobile browser currently being supported by the Android platform.

```
SELECT *
FROM <http://purl.org/carfo/example>
WHERE {
  ?browser a cou:SoftwareAgentModel ;
    dct:isVersionOf <http://purl.org/carfo/example/browser/android> ;
    rdfs:label ?label ;
    nie:version ?version .
}
```

[30] http://ontorule-project.eu/parrot
[31] Prefixes declaration has been omitted to improve queries readability

4.4.2 Query 2

Query 2 gets the attributes (e.g. full name, user name, group etc.) of those people who are also professors. This information was taken from Alexandru Ioan Cuza University of IASI, Romania[32] solely for testing purposes. However, in CKB, the real information related to the *User* aspect might be stored and tested as follows:

```
SELECT *
FROM <http://purl.org/carfo/example>
WHERE {
  <http://students.info.uaic.ro/people/professors> a foaf:Group ;
    foaf:member ?member .
  ?member a foaf:Person ;
    foaf:name ?fullname ;
    foaf:nick ?username .
}
```

4.4.3 Query 3

Query 3 gets a list of current browsers, which support both HTML5 Web Sockets and CSS3 Border Radius.

```
SELECT *
FROM <http://purl.org/carfo/example>
WHERE {
  ?browser a cou:SoftwareAgentModel ;
    rdfs:label ?label ;
    nie:version ?version .
  {
    ?browser cou:features <http://purl.org/carfo/example/feature/caniuse/css3/border-radius> .
  } UNION {
    ?browser cou:features <http://purl.org/carfo/example/feature/caniuse/js-api/websockets> .
  }
}
```

4.4.4 Query 4

Query 4 is performed on an FOAF knowledge base to find if two instances of `foaf:Person` are linked by `foaf:knows` relation. The information was extracted from FOAF files, collected by University of Maryland, Baltimore, USA[33]. Just like Query 1, this query was also made solely for testing purposes. The data represented contained 7118 FOAF documents collected from 2044 sites (identified by their symbolic IP address). A total of 201,612 RDF triples with provenance information were created.

```
SELECT *
FROM <http://purl.org/carfo/example>
WHERE {
  ?per1 foaf:mbox ?mailbox_1 ;
    rdf:type foaf:Person ;
    foaf:knows ?per2 ;
    foaf:mbox ?mail_box_2 ;
    rdf:type foaf:Person ;
    foaf:name ?pName_1 ;
    foaf:name ?pName_2 .
}
```

4.4.5 Query 5

Query 5 returns the number of features generated after the transformation process.

```
SELECT COUNT (*)
FROM <http://purl.org/carfo/example>
WHERE { ?s a http://purl.org/carfo/cou#Feature }
```

[32] http://students.info.uaic.ro/people
[33] http://ebiquity.umbc.edu/resource/html/id/82/

4.5 Visualizing statistical data from CKB

As mentioned in Section 3.3.2, some statistics provided by caniuse.com has also been translated to RDF graphs using the RDF Data Cube Vocabulary[34], the vocabulary to represent multidimensional data, which is under a W3C standardization process. The aim of this complementary transformation is to exploit the information with data-exploratory tools, such as visualization ones. For instance, these statistical information is available as a Tabels project[35], which enables end-users to interactively consume RDF data through graphical interfaces.

For instance, Figure 14 - Visualization data from RDF Data Cube shows the global percentage of use of each browser (obtained after adding the percentage of each browser-version instance usage).

Figure 14 - Visualization data from RDF Data Cube

4.6 Quill access approach

Another approach to providing access to the ontology would be via a scripting API using an interface that is compiled from the ontology. Such an approach would be applicable to the Quill[36] browser based authoring tool. This embodies a constraint-based expert system that generates the concrete UI for target platforms from the models for the domain, tasks and context of use. Further work would be needed on the means to compile the ontology into the interface definition language used by Quill. Although Quill is being developed as a design-time authoring tool, its reasoning engine could also be used at runtime to regenerate the user interface to match changes in the context of use, including constraints provided by the end user. Note that this is very much work in progress and would not be completed until near the end of the SERENOA project, or beyond the scope of the project.

[34] http://www.w3.org/TR/vocab-data-cube/
[35] http://idi.fundacionctic.org/tabels/project/serenoa/
[36] http://kt.abdn.ac.uk/wiki/Quill

4.7 Warehouse Picking Scenario and Role of CARFO Knowledge Base

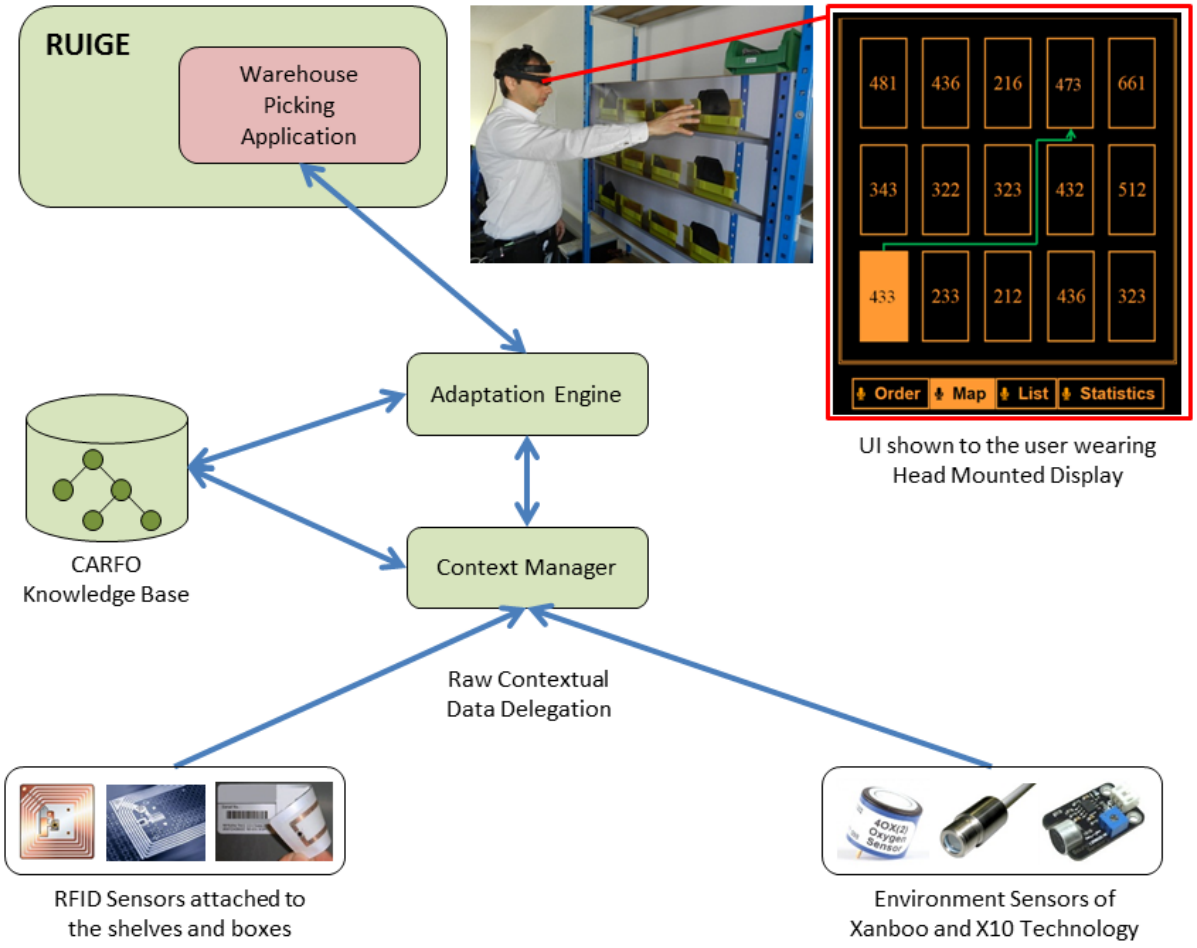

Figure 15 - Warehouse Picking Scenario and Role of CARFO Knowledge Base

Figure 15 - Warehouse Picking Scenario and Role of CARFO Knowledge Base depicts the warehouse picking scenario with different modules of SERENOA framework, particularly the interaction of Context Manager (CM) and Adaptation Engine (AE) with the CARFO Knowledge Base (CKB). Although there are different situations, which could arise to the picker during the picking process, in this section we are concentrating at only one case of *Traffic Jam situation*, where multiple workers are expected to approach the same path at the same time. The warehouse picking application adapts itself, with the help of adaptation rules and the knowledge stored in CKB, to minimise the risk of workers to wait for other people before picking the items and offering the picker with an alternative path to the desired shelves in order to complete the order.

In this particular case, the raw contextual data is retrieved and delegated towards the CM from the location sensors (in this case, the RFID tags attached to the shelves and their boxes). The raw contextual data consists of location (room, floor) and ID of the shelf (in this case, Nr. 433), hence the location of the picker as well. Using the inferred knowledge (with subsumption) stored in CKB, the CM determines the exact location of the picker (i.e. the building or factory area the picker is standing in). Rather a new UI generated at run-time by the RUIGE, the AE adapts the UI using a green line showing the shortest alternative path to the picker to the next shelf (Nr. 473) for the order completion. The blocked path is also showed to the picker with a red line, which is not depicted in Figure 15 - Warehouse Picking Scenario and Role of CARFO Knowledge Base.

5 Conclusions

In this deliverable it is presented the final version of CARFO as an OWL2 ontology in the shape of a Context of Use module for SFE adaptations. The deliverable covers ontological assumptions, design decisions and the explanation of main classes and properties of CARFO. This work relies on the idea that an ontology-based knowledge representation system facilitates SFE adaptation processes, providing further capabilities to the stack already available with XML. Ontologies can be used to specify the semantics of resources shared across systems. On one hand, the semantics defined with an XML schema are only available to the people that have specified it; while, on the other hand, the semantics defined using ontologies can be determined automatically by the systems at runtime (i.e., OWL semantics is based on Description Logics). Therefore, the use of ontologies can address the lack of data modelling in current adaptive SFE systems. They can be used to share and reuse knowledge, and on the basis of the semantics formally specified in the accompanied ontologies, they can make sense of the information needed for a given adaptation.

In the SERENOA project, CARFO is capable of capturing information about the user, the task, the system, the environment, and/or various aspects of the content (structure and presentation). This maximizes the amount of contextual information that can be used to accomplish sophisticated adaptation. Moreover, current adaptive service front-end systems rely on their own formalism and vocabulary for data representation. By the use of this ontology, the systems can share and reuse model information to solve the inherent lack of data that hinders sophisticated adaptations.

OWL ontologies are key requirements for building context-aware distributed systems defining the context information directly obtained from the context providers (e.g. sensors). The SERENOA platform provides an architecture for the run-time UIs generation/adaptation based on the CARFO ontology models and the CARFO Knowledge Base (CKB), generated as part of this deliverable. Furthermore, we have chosen an OWL-based context modelling (comprising User, Platform and Environment) because of the fact that ontologies are expressive, flexible data models that satisfy web-interoperability requirements based on the URI mechanism to identify resources and HTTP protocol to publish/retrieve data.

The ontology population process has been achieved in an automatic way using data sources such as caniuse.com, CTIC Device Description Repository (DDR) and Nokia Developer site. The number of generated triples has been almost duplicated from the previous experiment. Also, a successful proof of concept of the generated knowledge visualization has been carried out, using RDF Data Cube compliant tools.

Finally, the CARFO ontology is available for both human and machine consumption at http://purl.org/carfo and all the generated RDF triples are available from an SPARQL endpoint at http://data.ctic.es/sparql (at the named graph http://purl.org/carfo/example), showing the data results of this deliverable.

6 References

Berrueta, D., & Phipps, J. (2008). *Best Practice Recipes for Publishing RDF Vocabularies.* World Wide Web Consortium.

Breslin, J. G. (2005). Towards Semantically-Interlinked Online Communities. (A. G.-P. Euzenat, Ed.) *The Semantic Web Research and Applications* , 500-514.

Brickley, D. M. (2010). FOAF Vocabulary Specification 0.98. En D. B. Miller (Ed.), *Document 9 August 2010 Marco Polo Edition.*

Brossard, A., Abed, M., & Kolski, C. (2011). Taking context into account in conceptual models using a Model Driven Engineering approach. *Information and Software Technolog , 53* (12), 1349-1369.

Butler, M. H. (2002). CC/PP and UAProf: Issues, improvements and future directions. *Proceedings of W3C Delivery Context Workshop DIWS 2002.*

Butler, M. H. (2002). *Some questions and answers on CC/PP and UAProf.* HP Technical Report.

Coutaz, J. R. (2002). Foundations for a Theory of Contextors. *Proceedings of 4th International Conference of Computer-Aided Design of User Interfaces CADUI'2002 (Valenciennes, 15-17 May 2002)* (págs. 13-33). Kluwer Academics, Dordrecht.

Forum, W. (2001). *WAG UAProf.* Open Mobile Alliance.

Gergič, J. (2008). Addressing On-Demand Assembly and Adaptation Using a Runtime Intentional Versioning Engine. *Charles University in Prague* .

Group, W. O. (2012). *OWL2 Web Ontology Language: Document Overview.* World Wide Web Consortium.

Heath, T. &. (2011). Linked Data: Evolving the Web into a Global Data Space. En E. J. Hendler & F. V. Harmelen (Ed.), *Proceeding of the 17th international conference on World Wide Web WWW 08 . 1*, pág. 1265. Morgan & Claypool.

Indulska, J. R. (2003). Experiences in Using CC/PP in Context-Aware Systems. *Mobile Data Management* , 247-261.

Kießling, W. (2002). Foundations of preferences in database systems. *VLDB '02 Proceedings of the 28th international conference on Very Large Data Bases* (págs. 311-322). Morgan Kaufmann.

Klyne, G. R. (2004). *Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0.* World Wide Web Consortium.

Petasis, G., Karkaletsis, V., & Paliouras, G. (2007). *D4.3: Ontology Population and Enrichment: State of the Art.* BOEMIE Project FP-027538.

Prud'hommeauc, E. S. (2008). *SPARQL Query Language for RDF -- W3C Recommendation.* World Wide Web Consortium.

Rabin, J., Cantera Fonseca, J. M., Hanrahan, R., & Marin, I. (2008). *DDR Simple API.* W3C .

Rabin, J., Trasatti, A., & Hanrahan, R. (2008). *Device Description Repository Core Vocabulary.* W3C.

Schwinger, W. a. (2006). Modeling Web applications. En H. N. John Wiley (Ed.), *Web Engineering: Systematic Development of Web Applications.* (págs. pp. 39–64.). In G. Kappel, B. Pröll, S. Reich, and W. Retschitzegger, eds.

Tejo-Alonso, C., Berrueta, D., Polo, L., & Fernández, S. (2012). Current practices and perspectives for metadata on web ontologies and rules. *International Journal of Metadata, Semantics and Ontologies (IJMSO) , 7* (2), 93-100.

Tsinaraki, C., & Christodoulakis, S. (2005). Semantic user preference descriptions in MPEG-7/21. *Hellenistic Data Management Symposium HDMS.*

Wang, X. H., Gu, T., & Pung, H. K. Ontology based context modeling and reasoning using OWL. En G. T. Z. Da Qing (Ed.), *Proceedings of the Second IEEE Annual Conference on Pervasive Computing and*

Communications Workshops 2004 (págs. 18-22). IEEE.

Zimmermann, A., Lorenz, A., & Oppermann, R. (2007). An operational definition of context. *Modeling and Using Context , 4635*, 558-571.

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

- **TID:** Telefónica I+D (partner name)
- **UCL**: Université Catholique de Louvain (partner name)
- **ISTI**: Consiglio Nazionale delle Ricerche (partner name)
- **SAP**: SAP AG (partner name)
- **W3C**: Geie Ercim (partner name)
- **W4**: W4 (partner name)
- **CTIC**: Fundación CTIC – Centro tecnológico para el desarrollo en Asturias de las Tecnologías de la Información (partner name)

- **AAL:** Advanced Adaptation logic
- **Adaptability**: The capacity of a UI to adapt its behaviour through explicit human intervention.
- **Adaptable** User Interface: A UI that supports adaptability.
- **Adaptivity**: The capacity of a UI to adapt without any explicit human intervention.
- **API:** Application Programmers' Interface
- **AUI** / **Abstract User Interface**: A description of a UI that is independent from the specific UI resources available on the target computing platform.
- **CAA**: Context-aware Adaptation
- **CADS**: Context-Aware Design Space
- **CARF**: Context-Aware Reference Framework
- **CARFO**: CARF Ontology; an ontology that provides concepts for the representation of all knowledge in SERENOA's domain.
- **CKB**: CARFO Knowledge Base
- **Context**: A context identifies the situation in which e.g. a certain action occurs. Several aspects can be considered within the context: the user, the device, the environment…
- **CUI / Concrete User Interface**: A description of a UI that is dependent on the specific UI resources and modalities available on the target computing platform.
- **DAML+OIL:** successor language to DAML and OIL that combines features of both. Superseded by Web Ontology Language (OWL).
- **DL** / **Description Logic**: A family of formal knowledge representation languages. It is more expressive than propositional logic but has more efficient decision problems than first-order predicate logic. It is used in artificial intelligence for formal reasoning on the concepts of an application domain (known as terminological knowledge). It is of particular importance in providing a logical formalism for ontologies and the Semantic Web.
- **First Order Logic**: Formal logical system used in mathematics, philosophy, linguistics, and computer science. Also referred to as first-order predicate calculus, the lower predicate calculus, quantification theory, and predicate logic. It is distinguished from propositional logic by its use of quantifiers.
- **FUI / Final User Interface**: The UI produced at the implementation level, expressed as source code.

- **GUI**: Graphical User Interface: "In computing a graphical user interface (GUI, sometimes pronounced gooey) is a type of user interface that allows users to interact with electronic devices with images rather than text commands. … A GUI represents the information and actions available to a user through graphical icons and visual indicators such as secondary notation, as opposed to text-based interfaces, typed command labels or text navigation." [37]
- **HCI**: Human Computer Interaction: "Human–computer interaction (HCI) is the study, planning and design of the interaction between people (users) and computers. It is often regarded as the intersection of computer science, behavioural sciences, design and several other fields of study." [38]
- **ICT**: Information and Communication Technologies
- **IDE**: Integrated Development Environment: "An integrated development environment (IDE) also known as integrated design environment or integrated debugging environment is a software application that provides comprehensive facilities to computer programmers for software development." [39]
- **Interactor**: A single interaction object
- **MDA**: Model-Driven Architecture: the population of the software development process with difference models, each representing a particular view on the system being built
- **MDE**: Model-Driven Engineering
- **MEP**: Member European Parliament
- **Mereology**: the theory of parthood relations such of the relations of part to whole and the relations of part to part within a whole
- **Meta-UI**: Interactive system whose set of functions is necessary and sufficient to control and evaluate the state of an interactive ambient space
- **Ontology**: Formal and explicit specification of known concepts
- **OWL**: Web Ontology Language, W3C standard for ontology representation
- **Platform**: A class of devices that share the same characteristics in terms of interaction resources. Examples of platforms are the graphical desktop, PDAs, mobile phones, vocal systems…
- **QoSFE**: Quality of Front-End Services
- **RDF:** Family of World Wide Web Consortium (W3C) specifications originally designed as a metadata data model. Used as a general method for conceptual description or modelling of information that is implemented in web resources, using a variety of syntax formats.
- **Re-Molding**: Exploiting different modalities
- **SFE**: Service Front-End: Service front-ends are interfaces between the user and the backend through which the user gets access to various services.
- **SOA**: Service-oriented Architecture, A service-oriented architecture is essentially a collection of services. These services communicate with each other. The communication can involve either simple data passing or it could involve two or more services coordinating some activity. [40]
- **UI**: User Interface: "In the industrial design field of human–machine interaction, the user interface is the space where interaction between humans and machines occurs. The goal of interaction between a human and a machine at the user interface is effective operation and control of the machine, and feedback from the machine which aids the operator in making operational decisions."[41]

[37] Wikipedia online cited: 03 February 2011. http://en.wikipedia.org/wiki/Graphical_user_interface
[38] Wikipedia online cited: 03 February 2011. http://en.wikipedia.org/wiki/Human–computer_interaction
[39] Wikipedia online cited: 03 February 2011.http://en.wikipedia.org/wiki/Integrated_development_environment
[40] http://www.service-architecture.com/web-services/articles/service-oriented_architecture_soa_definition.html
[41] Wikipedia online cited: 03 February 2011.http://en.wikipedia.org/wiki/User_interface

- **XQuery**: W3C sanctioned query and functional programming language that is designed to query collections of XML data.