NEXOF-RA

NESSI Open Framework – Reference Architecture

IST- FP7-216446

Deliverable D1.1c

Advanced User-Service Interactions

(Contributions to Model and Specifications)

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

Due date of deliverable: 15/06/2010

Actual submission date: 15/06/2010

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

Change History

Version	Date	Status	Author (Partner)	Description
1.0	15/06/2010	draft	TID	Initial Version
1.1	30/06/2010	final	TID	Final version after revision

EXECUTIVE SUMMARY

This document captures the outcome of the work performed by WP1 “Advanced User-Service Interaction”, which is the result of a direct collaboration with the contributing NESSI Strategic and Compliant projects, other relevant projects (UsiXML [USIXML], FAST [FAST], Open [OPEN) and the external contribution gathered through the Open Process.

In summary, the work has focused on the development of the strategic vision, conceptual models and patterns for the Presentation concern of NEXOF-RA. Particularly, the work has consisted of collecting, unifying and consolidating the vision, results, input and contributions coming from:

- NESSI Strategic Projects (EzWeb)
- NESSI Compliant Projects (MyMobileWeb, FAST)
- other contributing projects that have also participated in the Investigation Teams, namely UsiXML [USIXML] and Open [OPEN]
- members of the Open Alliance on Service Front-Ends [SFE OA].

One of the most important achievements have been building consensus on a common vision and rationale, addressing all the concerned NEXOF-RA requirements. This vision is based on the following guiding principles, advocated by the aforementioned community of contributors and NESSI itself:

- **End-Users Empowerment and knowledge exploitation** which brings Web 2.0 concepts and paradigms to the Internet of Services arena.
- **Seamless Context-Aware User-Service Interaction** which advocates the introduction of context-sensitive user interfaces for the Internet of Services.
- **Universal Collaborative Business Ecosystems** which advocates the adoption of new service front-ends paradigms at business level.

The other major achievement is a set of patterns intended to realize our vision and aimed to contribute to the NEXOF Reference Architecture:

- **Front-End in E-SOA**. This pattern focuses on the functionalities provided by software components that are responsible for the design and execution of artefacts devoted to the NEXOF presentation
- **Mashup as a service**. In the context of the Internet of Services, this pattern suggests an architecture intended to enable end-user composition of heterogeneous Service Front-End Resources, yielding to Service Front-End Mashups.
- **Context of Use Management**. This pattern describes the architecture of a component able to detect, represent and provide the information about the Context of Use (User, Platform and Environment) for SFEs.

Document Information

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

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

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

Internal reviewers	Vadim Igorevich (TIE)		
	Pascal Bisson(Thales)		
AUTHORS (PARTNER)	N. Tsouroulas (TID), Jose M. Cantera (TID), José L. Díaz (TID)		
Responsible Author	José M. Cantera (TID)	**Email**	jmcf@tid.es
	Partner	**Phone**	

TABLE OF CONTENTS

1 Introduction

1.1 Scope of the deliverable

This document captures the outcome of the work performed by the “Advanced user-service interaction” WP, which has performed the work of developing the conceptual models and specifications for the Presentation concern of NEXOF-RA.

1.2 Objective of the work

This work package worked on the Service Front-End concern for a service-oriented architecture. Thereby it provides a baseline for models, formalisms, reference architectures, and specifications for building the front-end to access services that enable advanced, adaptive (and adaptable) user interfaces.

This work package covers the integration of formalisms and technologies to enable:

- Context-aware and personalized user-service interactions based on:
    - The user’s model (goals, interests, knowledge, social, personal, cognitive or behavioral aspects)
    - The situation in the interacting environment (location, date, time, …)
    - A computing platform (Delivery Context) (device, User Agent, network…)
- Flexible service customization by means of the selection, adaptation, and composition of the user interface by the end user

This work package is the NEXOF-RA instrument for tackling the above issues, from the point of view of service developers and end-users. The state-of-the-art analysis in the area of user interfaces shows that in this field there are a number of ad hoc and limited solutions exist in practice that are

- Focused only on specific platforms, users or contexts. (For example Web Access only from a desktop or laptop computer)
- Monolithic approaches that prevent customization or adaptation to the rapidly changing user needs and requirements

WP1 focused on the creation of a comprehensive and coherent solution to these problems, seeking feedback, collaboration and contribution targeted to create a common vision on advanced user-service interaction.

1.3 Methodology

The approach was to build the strategic vision and a set of descriptions for standards, glossary, formalisms, components, and patterns, at various levels of abstraction. These descriptions were then used as templates and guidelines for the construction of the entire architecture specification in a coherent and consistent manner.

A Functional Pattern provides an architectural solution to a software system that is mainly characterized (constrained) by its functionality requirements, i.e. the functionalities it must provide. (See Deliverable D7.2b Definition of an architectural framework & principles)

As a consequence, the proposed patterns for WP1 are functional patterns that provide architectural solutions to the set of functional requirements (problem).

These patterns identify the functional components, define the relationships among those components, and describe the functional architecture of the components related to the presentation concern.

The strategic vision was created from inputs from NESSI Strategic Projects and NESSI Compliant Projects (through direct interaction on a day-by-day basis), as well as other relevant projects (UsiXML [USIXML], FAST [FAST], Open [OPEN) with the help of the investigation teams of the 2 Calls we launched.

In the investigation teams we worked on some concrete areas of the SFE concern not covered completely by the NSP and NCPs in order to extend and complement the strategic vision with detailed specifications on key topics, and with the objective of starting the standardisation process on some of them (See 3 Contribution from the open process).

This strategic vision was also contributed to and adopted by the SFE Open Alliance (See 3.4.1 The SFE Open Alliance). This way we are leveraging the common vision established in NEXOF-RA, through the Open Alliance on Service Front-Ends that represent a wider group of projects than the contributing NESSI NSPs and NCPs.

1.4 Value of the result

The set of proposed patterns gives a broad vision of the components and functionalities related to the User Interfaces in traditional SOA architectures and in the Internet of Services architectures.

- The Front-End in E-SOA pattern focuses on the functionalities provided by software components that are responsible for the design and execution of artefacts devoted to the NEXOF presentation concern.
- The Mashup as a Service pattern provides the architecture of the components belonging to User Interface and Service Execution concerns of the Internet of Services.
- The Context of Use pattern defines the architecture of a software infrastructure for managing the *Context of Use* of an interactive system.

Standardization activities (related to the Presentation Concern) have been pursued to promote NEXOF-RA results in the area of open standards and to ensure that the necessities of NEXOF-RA are covered by the new standards developed. The W3C Model Based UI incubator group has taken the contributions of WP1 as inputs for the workshop celebrated on 04 May 2010 [MBUI].

2 INTRODUCTION AND FOUNDATIONS

2.1 Introduction

Service Oriented Architectures (SOA) have attracted a great deal of interest during the last years. In fact SOA increase asset reuse, reduce integration expenses and improve businesses agility in responding to new demands.

Nonetheless, until now, the mainstream development and research around SOA have been focused mainly on middleware and scalability, service engineering and automating service composition using BPM technologies. Little or no attention has been put on service front-ends which are, in our vision, a fundamental part of SOA. As a consequence SOA remain on a technical layer hidden to the end-user.

The evolution of web-based interfaces is a testimony of the progress achieved to improve service usability. However, existing, web-based service front-ends, are far from completely meeting end-user expectations. Most applications and information portals nowadays are still based on monolithic, inflexible, non-context-aware, non-customizable and unfriendly UIs. As a consequence end-users do not really benefit from the advantages promoted by service orientation in terms of modularity, flexibility and composition. In addition service front-ends are constructed in an ad-hoc manner and with the absence of formal engineering methods and tools that could accelerate the time to market.

NESSI and many other research projects have been working in areas related with this problem, each one addressing some specific issue and none the whole problem. Therefore we worked closely with the NESSI Strategic Projects and Open Alliance on Service Front-Ends to formulate an initial common vision for the NEXOF-RA reference architecture that applies the technologies and approaches developed in those projects and to the specific requirements defined within the NEXOF-RA project (WP8).

This chapter presents in detail this holistic vision of the next generation, user-centric service front-end technology. We present a series of guiding principles which promote the adoption of SOA front-ends that empower end-users and fully exploit Context and Knowledge. Our proposal will make end-users appreciate the benefits of SOA, fostering composition, loose coupling and reuse on the front-end layer.

2.2 Problem Statement

In recent years business IT systems and information portals on the Internet have increased their complexity to adapt to the demanding, and at the same time volatile, business and end-user requirements. This complexity has derived in an enormous effort to integrate, adapt and evolve applications and contents in order to satisfy user needs and expectations. The final results are severe penalizations in time to market with a subsequent big business impact.

Service Oriented Architectures (SOAs), which enable a “loose coupling” between applications, business logic and contents are starting to be used to alleviate these problems. Nonetheless, SOAs remain focused on service to service communications and automating service composition (using BPM technologies) on the backend layer.

SOA front-ends are still based on monolithic and non-customizable user interfaces and portals that invoke, when needed and in ad-hoc manner, backend services and processes. In this scenario services do not feature a “face” to human users, as they reside on a merely technical layer.

Existing approaches such as Internet mash-ups are valid in the short term, but unsustainable in the long term. The main issue is the restricted functionality of the UI which is limited to the combination of simple content rather than application functionality. High dependency on the underlying computing infrastructure is also a limiting factor. Changes in the original wrapped applications, in the back-end services, or in the portal infrastructure may cause a failure in the UI and render user service interaction unusable.

The reality is that nowadays users cannot actually benefit from the potential advantages provided by SOA in terms of composition, flexibility, customization and adaptability in a contextual manner. We advocate a next generation SOA front-end technology where users should be able to create their own applications by combining different pieces without any help from IT experts or deep knowledge about the underlying infrastructure. There are very good reasons to do so:

- IT systems and Internet applications nowadays are still designed and implemented taking as a reference the most common use cases and situations. Catering for more specific needs is, in many cases, commercially not viable. When it is, it almost always leads to increased complexity of the application and the interface, leaving less proficient users out.
- The traditional specify, design, implement, test and deploy lifecycle is no longer sufficient to capture the rapidly changing user needs and requirements. Users are increasingly dependent on IT systems and, at the same time, more knowledgeable about the problems they want to solve and how to tackle them. They hold a very valuable knowledge and some of them are willing to devise new solutions if they are given the means to do so. Traditional tools and methods raise the bar too high for non-IT-aware users in order to be able to contribute new solutions.

These are well recognised Web 2.0 principles that are still not adequately supported by current service front-ends and engineering tools. Providing solutions tailored to users needs and allowing them to contribute their knowledge and creativity represents a huge potential yet to be unleashed. Service front-end is the best place to do so, since it is the layer that imposes the least IT knowledge requirements on the user and is the one closest to the user and the problem to be solved.

Therefore, it is necessary to devise a new SOA front-end capable to provide an environment which empowers end-users to easily use, customize, contribute, enhance and compose services coming from different providers. Furthermore, users should be able to create their own applications by combining different pieces without any help from IT experts or deep knowledge about the underlying infrastructure.

In addition the new generation SOA front-end should embrace the Web 2.0 principles, exploiting user's collective intelligence and domain knowledge. Service front-ends should be in charge of knowledge capture and further exploitation, as they are the architectural components that deal with user interaction. Thus, new challenges appear with respect to user's knowledge gathering, representation and integration with formal knowledge. At this respect an important research topic is the seamless integration between ontologies (formal semantics) and folksonomies (light semantics).

Once the user knowledge has been properly extracted it should be exploited to enrich context-awareness, to personalize the user-service interaction and to improve existing processes or derive new ones.

From the engineering point of view, the development of context-aware service front-ends is insufficiently supported by existing methodologies and tools. Instead, user interfaces are designed manually for service interfaces and business processes. This fact calls for the need to develop new technologies and tools that simplify and accelerate the development of context-aware service front-ends. Such front-ends will adapt seamlessly to the target environment, providing a harmonized user experience in every device, network or situation.

From the business and enterprise applications perspective, the adoption of new generation service front-ends is also challenging. Traditional monolithic user interfaces will be phased out by advanced environments based on services front-end composition and customization. In addition employees will start to be seen as an essential piece of systems. Tools and technologies for exploiting collective knowledge will start to appear. Last but not least, engineering departments will need to incorporate new paradigms supporting the development of applications that adapt seamlessly to each environment and situation.

2.3 Guiding Principles

From our point of view, the challenges explained in the previous section call for the need to user-centric SOAs based on a new generation of service front-end technologies. Such technologies will enable the massive deployment of services on the Internet, driven by the following guiding principles:

- **End-Users Empowerment,** enhancing traditional user-service interaction by facilitating the selection, creation, composition, customization, re-use and sharing of applications in a personalized operating environment. This principle has been introduced by the NSP, EzWeb.
- **Seamless Context-Aware User–Service Interaction**. New generation service front-ends should have the capability to detect contextual information, to represent it, to manipulate it, and to use it to adapt seamlessly to each situation, supporting human users in a more

effective, personalized and consistent way. Novel engineering tools and methods should be devised in order to support context-aware service front-ends. This principle is advocated by the NCP, MyMobileWeb.

- **End-Users Knowledge Exploitation** This principle aims to exploit users' domain knowledge and collective intelligence to improve service front-ends. End-Users knowledge can be used to tag resources using light semantics, to assist while interacting with services, to enrich contextual information (for example by means of automatic user profiling) and to infer new candidate processes to be later automated (on the backend). This principle is being pursued by the EzWeb NSP.
- **Universal Collaborative Business Ecosystems**. Enterprise systems should incorporate advanced user-centric, context-aware front-ends to enable their employees and other stakeholders to exploit, and share their extensive domain expertise, and their thorough business knowledge. Employees, customers, developers and providers will collaborate to create and improve enterprise applications, sharing, reusing, changing and combining existing context-aware components (services, contents, things...). This principle is aligned with the EzWeb vision.

2.4 Enabling users to author and share resources

The "End-User Empowerment" principle gives an active role to end-users letting them author, enhance, share and customize their own applications and operating environment (workspace), thus going beyond traditional, monolithic user-service interaction. Figure 1 depicts our technical proposal (implemented by the EzWeb NSP) to materialize such principle. The main component of the underlying platform is a resource catalogue containing applications (gadgets) that implement the user interface that allows end-users to access one or more services. This resource catalogue plays a similar role than backend service and process registries.

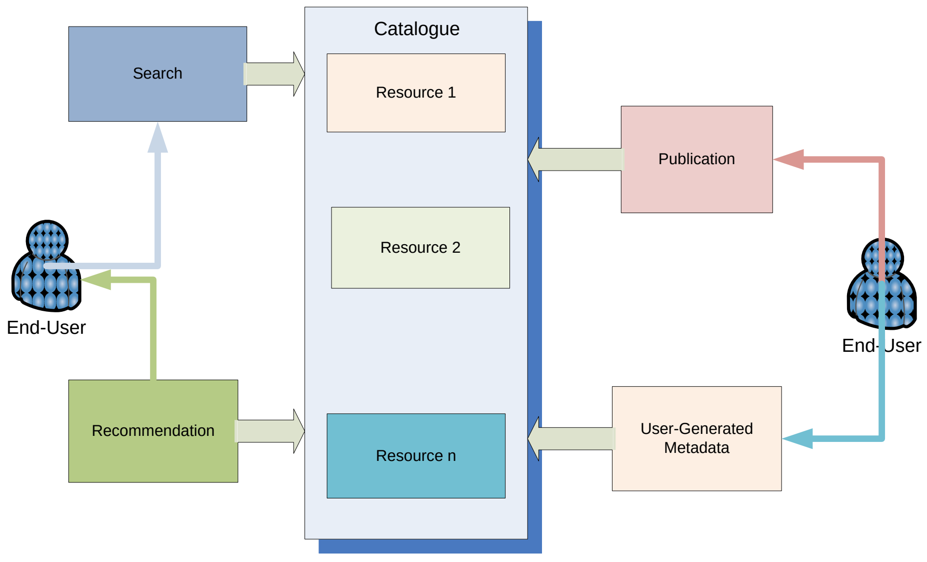


Figure 1 Resource Catalogue

Our proposed catalogue will be responsible for maintaining the information about each front-end (gadget) and associated metadata (template descriptors, deployment information, author, icons, versions, formal-semantics-based annotations, user-assigned tags and punctuation…).

2.5 A Framework for Context-Aware Service Front-Ends

The "Seamless Context-Aware User-Service Interaction" principle will drive the construction of novel service front-ends capable of using contextual information to influence their behaviour, thus supporting human users in a more effective and personalized way. This principle is aligned with the vision of the MyMobileWeb project.

Particularly, context-awareness will provide the following benefits to service front-ends:

- User Interface harmonization for every device or mode of interaction. Contextual information will enable automatic content and application delivery tailored to each target environment.
- User-Service interaction enhancement:
    - Contextual information can be exploited to present a different workspace depending on each situation. Each operating environment may include different functionalities or interaction modes depending on the situation (at home, at work, on vacation, on the move, on a business trip …).
    - Context can be used by resource catalogue search and retrieval modules to recommend the best gadgets to be used under a given situation, matching rich resource descriptions with the characteristics of the target environment.

To materialize our vision, new tools, formalisms and engineering methods need to be devised in order to simplify the development of context-aware service front-ends:

- A device and modality independent Declarative Authoring Language (and associated standard context-based adaptation policies). This is a fundamental piece for those service front-ends intended to work on multiples devices or modes of interaction. For example, MyMobileWeb provides the IDEAL2 language [IDEAL2] for this purpose. Other related languages are UsiXML [UsiXML] and MariaXML [MariaXML](developed by the FP7 Open Project [Open]).
- A Context Framework, as proposed by MyMobileWeb, composed, at least, by the following elements:
    - A Universal and Extensible Context Model enabling the development of Context Vocabularies.

- A Context Mediation Layer, including binding formalisms, hiding applications from the complexities of dealing with multiple and heterogeneous Context providers.
- A Universal Context API, providing a simple, uniform programming abstraction for the development of context-aware services.

Once this Context Framework is in place an application can infer which actions should be triggered, what data is relevant, and what topics are interesting for the user within a specific context.

2.6 User's knowledge capture and exploitation in service front-ends

The "End-Users Knowledge Exploitation" principle aims to exploit users' domain knowledge and collective intelligence for enhancing service front-ends. This principle can be materialized by:

- Encouraging users to create and compose their own service front-ends as a way to solve problems that were not anticipated and that require thorough domain knowledge. EzWeb is an enabler technology for doing so.
- Stimulating users to share their knowledge by means of semantic descriptions and annotations for service front-ends. This point is really challenging since it will imply the integration between folksonomies (light semantics) and ontologies (formal semantics). This is an outstanding challenge.
- Monitor users behaviour and input to service front-ends in order to:
    - Enrich contextual information about the user by employing *automatic user profiling*, to overcome the problem of incomplete or inaccurate user-provided profiles. Users with similar interests can also be grouped into classes, and assumptions about individual's interests can be made based on the interests of the whole group. This leads to the concept of social profiling.
    - Enhance interaction, for example current search engines use previous inputs to assist the user while searching. The same idea could be applied, for instance, to application forms.
    - Detect interaction or composition patterns in order to infer new processes suitable to be later automated (on the backend).

  We are (still) not in a position to solve these challenges.

All the novel approaches described above should be the starting point for a new innovation culture based on continuous improvement via reusing and sharing of knowledge. In fact, users will connect and use resources in their workspace according to their skills, situation or others. Wheel-reinventing in organizations or by individuals regarding service front-ends will no longer happen. Competition will rise as there will be an ecosystem of providers and consumers of resources willing to provide the best solution for a problem. Users and not marketing departments will evaluate, vote and choose the best resources for a problem.

2.7 Collaborative, user-centric and context-aware IT systems

The “Universal Business Ecosystems” principle aims at the incorporation of advanced user-centric, context-aware business front-ends to enable employees and other stakeholders to exploit, and share their extensive business expertise. This principle has been introduced by EzWeb.

As a consequence employees, customers, developers and providers will collaborate to create and improve enterprise applications, sharing, reusing, changing and combining existing context-aware components (services, contents, things...). O the other hand we acknowledge that such a new and innovative vision will pose a change that will need to be managed within businesses in order to be successful in the long term.

To materialize this principle it is necessary to explore alternatives that could be used in the development of enterprise applications and adopt a fully user-centric approach. This calls for the need to:

- Extend the concept of end-user to people who develop, use, market and exploit the system.
- Consider all user-roles as a fundamental part of systems, in order to develop innovative, comprehensive and coherent solutions to B2C and B2E solutions.
- Enterprise Collaboration Systems (ECS) evolution towards a new paradigm in which the more skilled employees should be co-producers not only of information, but also of software, services and applications.
- Empower collaboration without requiring any programming skills, only thorough business knowledge. The main aim will be both to foster the company’s competitive advantage without involving IT departments, and sharing of user-created solutions with the remainder of the organization.
- Promote a new innovation culture introducing new operating procedures by mixing and integrating available services.
- Break the frontiers between systems, companies and users in favour of a universal platform, the Internet of Services.

Embrace the Software as a Service (SaaS) model as an effective software-delivery mechanism. This will change the department’s focus from deploying and supporting applications to managing the services implemented by these applications.

2.8 Reference Architecture requirements and objectives

In the following table there is a relation between the system requirements for the presentation concern, and the proposed pattern we have developed. The overall strategic vision developed comply with this high-level requirements but go far beyond them and try to solve more specific challenges in the area of Service Front-Ends, as defined by the contributing projects, NESSI and the members of this work package.

Based on these requirements and the strategic vision developed, the proposed reference architecture will help system engineers implementing systems that provide functionalities explored by the presentation concern, and to implement the overall vision and satisfy the requirements defined within NEXOF-RA.

System Requirements	Pattern
SR 7. How can users be supported to access all the SOA-based system functionalities (Services and Processes creation, management, execution, etc.)?	Front-End in E-SOA
SR 1.4. How can a service be executed (run-time support)? SR 5.1. How can processes be designed in terms of the services they are composed of (Orchestration, Choreography descriptions)? SR 5.1.2. How can process be designed in order to manage interoperability matters (data mapping, message transformation)?	Mash up as a service

Table 1: System Requirements and Patterns

3 CONTRIBUTION FROM THE OPEN PROCESS

This section summarizes the contribution obtained from the open process, mainly the investigation teams, but also includes the contribution obtained from the collaboration with NESSI projects, from the collaboration from other relevant projects in the Presentation Concern, and the collaboration with the SFE Open Alliance.

With all this contribution and taking into account the requirements establish by the project, we compose a complete strategic vision for the presentation concern.

With this vision in mind, we have described a set of terms for glossary, and we have developed the selected patterns.

3.1 Activities related to the open process

3.1.1 The proposed Topics

The proposed topics for the investigation teams focused in the presentation concern are:

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

3.1.2 The Investigation teams

Table 2: IT List

WP10 System Requirements[1]	Investigation Team
SR 7. How can users be supported to access all the SOA-based system functionalities (Services and Processes creation, management, execution, etc.)? SR 1.4. How can a service be	Declarative Authoring Languages for User Interfaces

[1] D10.1 Requirements Report

executed (run-time support)? SR 5.1. How can processes be designed in terms of the services they are composed of (Orchestration, Choreography descriptions)? SR 5.1.2. How can process be designed in order to manage interoperability matters (data mapping, message transformation)?	
SR 7. How can users be supported to access all the SOA-based system functionalities (Services and Processes creation, management, execution, etc.)?	Context Model and Universal APIs

Table 3: IT Process Report

Investigation Team	start	end	Registered participant	Position paper	Effective participant	Meeting	Phone Call
Declarative Authoring Languages for User Interfaces	21-October-2008	1-March-2009	7	4	4	1	5
Context Model and Universal APIs	21-October-2008	1-March-2009	10	5	6	1	3

3.1.2.1 *Declarative Authoring Languages for User Interfaces*

Rationale

Identified system requirement on Model-Based Engineering Tools for context-sensitive and multi-target user interfaces.

Objectives

This IT was focused on declarative authoring languages suitable for specification of adaptable user interfaces for the ubiquitous web. The ultimate goal in mind was to achieve device independency and context sensitivity of the service front-end UI.

More abstract languages that go one step further and allow modality independent interface specifications were also be part of the investigation performed. The results could be used to define future strands of evolution of the selected language or languages.

Criteria to issue the call

The creation process consisted of:

- initial expression of interest.
- confirmation with a position paper submission.
- formal kick-off meeting held in Brussels (at the Commission Offices) on October 2008.

Every IT has a leader which is normally a person from the organization that it is leading the working package, thus Telefónica for WP1. It is important to note that the leading organization can also participate with other people that will play the role of contributors representing specific projects. This was actually the case for Call 1.

Last but not least, a web space (on the NEXOF-RA site) and mailing lists were created for supporting the work of the different launched ITs.

Placement in architecture

It is related to the Presentation Concern in NEXOF-RA.

Investigation team reports

Available at

http://forge.morfeo-project.org/wiki_en/index.php/Interactive_Application_Models

Topics

Declarative Authoring Languages for User Interfaces, either developed within a project or available as an open standard, to be adopted by NEXOF-RA.

Setup of the team

- Nikos Tsouroulas (TID, Spain)
  - IT Leader
- José Manuel Cantera (TID, Spain)
  - Representing Eureka-CELTIC MyMobileWeb
- Jose Luis Díaz (TID, Spain)
  - Representing Eureka-CELTIC MyMobileWeb
- Fabio Paternó (CNR, Italy)
  - Representing FP7 OPEN & ServFace
- Jean Vanderdonckt (UCL, Belgium)
  - Representing FP7 Human & UsiXML Consortium
- Juan Manuel González Calleros (UCL, Belgium)

- Representing FP7 Human & UsiXML Consortium
- Xabier García de Cortazar (Tekniker, Spain)
    - Representing FP6 WearIT@Work
- Miguel Jiménez (UPM, Spain) Representing NSP EzWeb

Summary of activities

- 5 conference calls (every two weeks approximately) to follow the advance of the IT.
- Kickoff physical meeting in Brussels at the Commission
- A Face to face meeting in UCL (Universite Catholique de Louvain, Belgium) (29th-30th January 2009).
- Usage of the IT mailing list user-interaction-it@nexof-ra.eu for discussions
- Usage of a wiki page for the initial input provided by the different members

  http://forge.morfeo-project.org/wiki_en/index.php/Interactive_Application_Models

Role assigned

TID was the IT leader

Process followed

Taking the initial papers, NSP and NCP as a starting point, we have evolved its ideas with the ones provided in the several meetings of this IT to produce the result of the main models for interactive applications.

Results

Identification of the main models for interactive applications: Task, Abstract User Interface [AUI], Concrete User Interface [CUI], Mapping, Behaviour, Domain and Creation of harmonized Tasks models and AUI models taking the different contributions as input

Innovation points

Identification of the main models for interactive applications: Task, [AUI], [CUI], Mapping, Behaviour, Domain

Impact on standardization

The members of the IT have agreed to continue the work under the **W3C Model-Based UI XG**, [MBUI] hence promoting the results in the area of open standards. NEXOF-RA AUI and CUI models will be contributed to the MBUI XG. Further work on standards will be pursued.

3.1.2.2 *Context Model and Universal APIs*

Rationale

Identified system requirements on Context-Sensitive service front-ends.

Objectives

This IT was intended to deal with the adoption of a Context Model for supporting context-sensitive front-ends. The following context facets were under the scope of the so-called IT:

- User Profile: global preferences, interests, skills and social network
- Delivery Context: device, network, user agent and local settings (font size, volume, brightness …)
- Environment: location and moment in time

The adopted Context Model was required to be extensible allowing other properties and aspects (standard or application-specific) to be included in the future.

In addition the IT also was looking for a Context API with the following functionalities:

- Platform independent
- Generic and extensible, allowing to work with different vocabularies of contextual properties
- It should support the notion of properties, aspects and components of the Context
- It should provide not only query-response functions but also publish and subscribe mechanisms for notifying contextual changes to applications

Criteria to issue the call

The creation process consisted of:

- initial expression of interest.
- confirmation with a position paper submission.
- formal kick-off meeting held in Brussels (at the Commission Offices) on October 2008.

Every IT has a leader which is normally a person from the organization that it is leading the working package, thus Telefónica for WP1. It is important to note that the leading organization can also participate with other people that will play the role of contributors representing specific projects. This was actually the case for Call 1.

Last but not least, a web space (on the NEXOF-RA site) and mailing lists were created for supporting the work of the different launched ITs.

Placement in architecture

It is related to the Presentation Concern in NEXOF-RA.

Investigation team reports

Available at

http://forge.morfeo-project.org/wiki_en/index.php/Interactive_Application_Models

Topics

Setup of the team

- Nikos Tsouroulas (TID, Spain)
  - IT Leader
- José Manuel Cantera (TID, Spain)
  - Representing Eureka-CELTIC MyMobileWeb & NSP EzWeb
- Jose Luis Diaz (TID, Spain)
  - Representing Eureka-CELTIC MyMobileWeb & NSP EzWeb
- Fabio Paternó (CNR, Italy)
  - Representing FP7 OPEN & ServFace
- Jean Vanderdonckt (UCL, Belgium)
  - Representing FP7 Human & UsiXML Consortium
- Juan Manuel González Calleros (UCL, Belgium)
  - Representing FP7 Human & UsiXML Consortium
- Miguel Jiménez (UPM, Spain)
  - Representing Eureka-CELTIC MyMobileWeb
- Lorant Farkas (Nokia Siemens Networks, Hungary)
  - SharME & Other internal NRC projects
- Carlos Juiz (UIB, Spain)
  - Representing project SOSA (National Funded Project)

Summary of activities

- 3 conference calls to follow the advance of the investigation team.
- kickoff physical meeting in Brussels at the Commission
- Use of the IT mailing list user-interaction-it@nexof-ra.eu

Role assigned

TID was the IT leader

Process followed

Taking the initial papers, NSP and NCP as an starting point, we have evolve its ideas with the ones provided in the several meetings of this IT to produce the result of the main models for interactive applications.

Results

Creation of harmonized model of Context taking the different contributions as input

Innovation points

Creation of harmonized model of Context taking the different contributions as input

Impact on standardization

Delivery Context Ontology [DCONTOLOGY], (W3C Working Draft 16 June 2009) aligned with the Delivery Context Model of NEXOF-RA. NEXOF-RA is having a leading role in this spec, be means of TID, with the support of other major IT players (Vodafone, AT&T, Volantis, IBM, …)

3.2 Collaboration with NESSI projects

This section gives an overview of the projects that has been contributed to the NEXOF-RA Presentation concern.

3.2.1 MyMobileWeb (Eureka-Celtic Project)

MyMobileWeb (Eureka CELTIC EU-CP04-20) is the open source reference implementation of the next generation content and application adaptation platform for the Mobile Web. MyMobileWeb provides the following technologies:

- IDEAL language for the (semantic) description of the device independent user interface
- Context ontologies and APIs
- Content and application adaptation runtime (semantics-based)
- Mobile AJAX Framework
- Content and service discovery
- Semantic Bar (enriches browsing exp.)
- Authoring tools (Eclipse Plug-in)

MyMobileWeb enables the creation (in time to market) of high quality mobile applications and contents capable of adapting to multiple Delivery Contexts, satisfying user and content providers' expectations. MyMobileWeb is intended to the development of high-quality dotMobi portals and Rich Mobile Web applications intended to work in multiple delivery contexts

3.2.2 EzWeb (NESSI Strategic Project)

The EzWeb project pursues the development of an enriched enterprise mash-up platform and the development of key technologies to be employed in building the **front end layer** of new **generation SOA architecture**. EzWeb aims at supporting the following criteria:

- **End-users must feel fully empowered**, They must be able to self-serve from a wide range of available resources, providing access to content and application services, in order to set up their own personalized operating environment in a highly flexible and dynamic way ("Do it yourself", IKEA philosophy).

- **Active participation of users has to be enabled**, allowing them to create resources as well as share and exchange both knowledge and resources with others and learn together, thus accelerating the way innovations and improvements in productivity are incorporated.
- **Interaction must be adapted and relevant to context**, giving the term "context" the widest possible meaning, in a way that comprises both user context (knowledge, profile, preferences, language, information about social networks the user belongs to, etc.) and delivery context (static and dynamic characteristics of the device used for access, geographical and time location, connection bandwidth, etc.). Dynamic context variability and user mobility must also be taken into consideration.

3.3 Collaboration with other relevant projects

3.3.1 UsiXML (ITEA2 Project)

UsiXML [USIXML] is an existing XML-compliant mark-up language that describes an UI for multiple contexts of use such as character, graphical, auditory or multimodal interfaces. UsiXML is intended for developers, as well as analysts, designers, human factors experts, etc.

Thanks to UsiXML, non-developers can shape the UI of any new interactive application by specifying it in UsiXML, without requiring the programming skills usually found in mark-up and programming languages. UsiXML is a declarative language capturing the essence of what an UI is, or should be, independently of physical characteristics. Currently UsiXML (only) supports interaction device independence, computing platform independence and interaction modality independence.

UsiXML has been recently labelled as an ITEA 2 Project (ITEA Call 4, 2008). In this new phase the project will define, validate, and standardize an open User Interface Description Language increasing productivity & reusability and improving usability & accessibility of industrial interactive applications by supporting the "μ7" concept: multi-device/user/linguality/organisation/context/modality/platform.

3.3.2 FAST (FP7 Call 1 Project)

FAST (FP7-216048) aims at providing an innovative visual programming environment that will facilitate the development of next-generation composite user interfaces. It constitutes a novel approach to application composition and business process definition from a top-down user-centric perspective. The main objective of the project is to create a new a visual programming environment that will facilitate the development of complex front-end gadgets, involving execution of relatively complex business processes that rely on back-end semantic Web services.

The approach adopted will be user-centric rather than program-centric: instead of first building programs that orchestrate available semantic Web services and then try to figure out how to implement interaction with the users at given points of the process execution flow, programmers will start from the front-end gadgets that the user will see and interact with, and then, afterwards, they will visually establish the connection to back-end Web services going through process execution flows if necessary. The way programmers will visually establish this connection will follow an approach similar to what sequence diagrams in UML look like.

3.3.3 OPEN (Open Pervasive Environments for migratory iNteractive Services) (FP7 Call 1 Project)

The main goal of the OPEN [OPEN] project is to provide a general and open migratory service platform solution based on a sound and innovative scientific approach developed by a multidisciplinary consortium combining the expertise of three technological world leaders, three well known research organisations and one SME (CNR-ISTI, AAU-CTIF, NEC Research Center, SAP AG, Vodafone Omnitel NV, Arcadia Design srl, Clausthal University of Technology).

The service platform will be able to interoperate with existing technologies by:

- adapting and preserving the state of the software application parts dedicated to interacting with end users
- supporting mechanisms for logic reconfiguration
- defining suitably flexible mechanisms from the underlying layers

In particular, the OPEN project provides integrated solutions able to address three aspects: device change, state persistence and content adaptation. This is obtained through a middleware able to consider and integrate various aspects: adapt and preserve the state of the software application parts dedicated to interacting with end users; support mechanisms for application logic reconfiguration; and identify suitably flexible mechanisms available in the underlying network layers. The user interface adaptation and continuity is obtained with the support of logical user interfaced descriptions.

The resulting middleware is able to interoperate with existing technologies. Thus, OPEN aims to offer an intelligent infrastructure able to: deliver seamless and transparent support to users in carrying out their tasks when changing available devices; provide and coordinate reliable, dynamically changing/reconfiguring services; offer personalised user interaction by exploiting different interaction modalities and network technologies through an infrastructure able to provide the necessary context information (e.g. available devices, connectivity, users and related transformations for content adaptation).

3.4 Open Alliances

Open Alliances have been defined as an instrument for the purpose of achieve cooperation between projects in the area of the future Internet of Services.

These projects are not necessarily linked to the same research programme (i.e., FP7, Eureka, national-level programs or private corporate programs.)

An Open Alliance is conceived by industrial and academic organizations that declare to share a concrete vision on how part of the infrastructure for the future Internet of Services can be implemented and together commit to spend resources to:

- Produce concrete open specifications of components in the Reference Architecture they envision, taking an active role in the NEXOF-RA process or similar processes worldwide that may be instrumental for the adoption of their envisioned Reference Architecture.
- Integrate results of projects they run and agree to link to the Open Alliance, with the goal of producing a complete and coherent set of software components that work as reference implementation for each part of the envisioned Reference Architecture. These components will ideally be part of NEXOF and some could be open source while others not, according to the Alliance's decisions.

In order to deliver on these goals, Open Alliances have some special characteristics that set them apart from other initiatives or collaborations schemes.

First of all, Open Alliances membership is open only to those who embrace a set of basic shared principles and vision set by existing members and agree on integrating already available (or future) results of their work with those coming from a number of base/founding projects that were brought in by founding members. Base/founding projects need to have results that can be used right from the beginning to form the reference implementation of base components of the envisioned Reference Architecture. A fundamental aspect of Open Alliances is that they are not limited to FP7 funded projects. Actually, any research project can contribute its results to an Open Alliance (i.e. funded by national programs, Eureka programs or funded privately). These rules are driven by pragmatism: cooperation can only be effective where there is a clear view on how integration of results can be materialized while, on the other hand, no result should be excluded because of their origin if it can be integrated.

Components produced within an Open Alliance have a lifecycle beyond those of the private or publicly funded projects supporting research and development activities during a given period of time. Members commit to follow a joint Roadmap that drives evolution of components in the Reference Architecture or the addition of new ones. The open source approach followed in development of some of these components ease evolution over time as well as collaboration between projects in different research programmes.

As opposed to NESSI Working groups, which are mostly focused on SRA development and research challenges definition and are generally open to everybody, Open Alliances are more focused on integration and delivery of results towards a coherent and complete suite of components materializing a particular architectural vision. FP7 Collaboration Working Groups (CWGs) shouldn't also be confused with Open Alliances. FP7 CWGs are a useful instrument to exchange knowledge between FP7 projects and, as such, they are open to all FP7 projects. However, they are limited to FP7 projects and cannot impose a concrete vision. On the other hand, they may stimulate incubation of Open Alliances.

3.4.1 The SFE Open Alliance

A first Open Alliance on Service Front-Ends was incubated during a number of NESSI WGs and FP7 Service Front End CWG meetings. Started in September 2008, now includes important industrial and academic members such as Telefónica I+D, SAP, Universidad Politécnica de Madrid, Vienna University of Technology, Institute of Information Science and Technologies and the Université Catholique de Louvainne. As its name suggests, the mission of this first Open Alliance is to deliver an architecture and reference implementation of the service front-end layer for the future Internet of Service. More information can be found on the alliance's web site: http://sfe.morfeo-project.com. Discussions about the creation of another Open Alliance on Cloud infrastructures are currently under way.

We believe that this focus on concrete parts of the Future Internet where there exists expertise, results, vision and commitment can allow Open Alliances to:

- play a relevant role in definition of software strategies in Europe
- be instrumental to integrate research results (FP7 and beyond)
- play a relevant role in the Future of Internet Assembly initiative
- contribute in key areas of NEXOF-RA and initiate and accelerate implementation of the main components of NEXOF
- help to pave the way from research projects to actual products that can be delivered to the market

3.5 Standardization Activities

This section describes the standardization activities (related to the Presentation Concern) in which we have participated and we are participating with the support of the NEXOF-RA Project.

The final aim of our participation is both to promote NEXOF-RA results in the area of open standards and to ensure that the necessities of NEXOF-RA are covered by the new standards developed. As a result there will be an alignment between NEXOF-RA and Open Standards which it is a strategic target for NESSI and NEXOF.

3.5.1 Ubiquitous Web Applications WG

The Ubiquitous Web Applications Working Group [W3C07] seeks to simplify the creation of distributed Web applications involving a wide diversity of devices, including desktop computers, office equipment, home media appliances, mobile devices (phones), physical sensors and effectors (including RFID and barcodes). This will be achieved by building upon existing work on device independent authoring and delivery contexts by the former Device Independent Working Group [W3C04], together with new work on remote eventing, device coordination and intent-based events.

With regards to NEXOF-RA the main standard that this group is developing is the Delivery Context Ontology (DCO). The DCO he Delivery Context Ontology provides a formal model of the characteristics of the environment in which devices interact with the Web or other services. The Delivery Context includes the characteristics of the Device, the software used to access the service and the Network providing the connection among others. Telefónica is leading such activities (as an editor of the specifications) in cooperation with other stakeholders, namely Mobile Aware, Volantis Systems (content adaptation solution providers), AT&T and Vodafone. One important achievement is that there is full alignment between the current draft of the "Delivery Context Ontology Recommendation" and the identified necessities (summarized by the requirements) of NEXOF-RA.

3.5.2 Model-Based UI XG

The activities carried out under this Incubator Group (XG) are aimed at promoting the results of NEXOF-RA in the area of standards for user interface declarative models and languages. This is stimulated by the fact that some members of the NEXOF-RA Investigation Team on Declarative User Interfaces (see the chapter on the Collaboration Process) are also members of the XG.

Our final target is to set up the initial steps towards the standardization of models and languages for representing user interfaces. NEXOF-RA results are demonstrating that such standardization is feasible. Hence, it will be worth to continue with further steps in order to devise new W3C Recommendations devoted to this matter.

A W3C Workshop on "Future Standards for MBUI" has been celebrated on 13-14 May 2010 in Rome [MBUI].

4 CONTRIBUTION TO THE REFERENCE ARCHITECTURE

4.1 Contribution to Guidelines and Principles

WP1 has contributed a set of terms to the glossary of WP6 on the presentation area. The detailed set of terms that WP1 has contributed to the glossary of WP6 on the presentation area is included in D1.1c Appendix A: Contribution to other WPs.

4.2 Contribution to the Reference Model

4.2.1 Introduction

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

The reference model proposed are mature and good quality contributions coming both from NESSI Strategic Projects and the cooperating projects that have participated in the Investigation Teams and the Open Construction Process (MyMobileWeb, EzWeb, UsiXML, FAST).

The whole set fundamental models for describing multi-target interactive applications that can be able to adapt to its Context of Use and the set of concepts and relationships is presented in the document D1.1c Appendix B: Investigation team contributions.

The following sections contain an introduction to the concepts of Model-Based UI, and the **CAMELEON Reference Framework** in particular.

The **rationale** for selecting the proposed models can be found in the results of the CAMELEON FP5 Project (FP5-IST4-2000-30104) and the *CAMELEON Unified Reference Framework*. CAMELEON [CAL03], describes a framework that serves as a reference for classifying user interfaces supporting multiple targets, or multiple contexts of use in the field of context-aware computing.

The detailed Contribution to the Reference Model on the presentation concern is included in D1.1c Appendix B - Investigation teams contributions.

4.2.2 Model-Based Approaches for User Interfaces

4.2.3 Introduction

The purpose of Model-Based Design is to identify high-level models, which allow designers to specify and analyse interactive software applications from a more semantic oriented level rather than starting immediately to address the implementation level. This allows them to concentrate on more important aspects without being immediately confused by many implementation details and then to have tools which update the implementation in order to be consistent with high-level choices. Thus, by using models which capture semantically meaningful aspects, designers can more easily manage the increasing complexity of interactive applications and analyse them both during their development and when they have to be modified [PAT05].

During more than a decade model-based approaches have evolved in parallel with the aim of coping with the different challenges raised by the design and development of UIs in continuously evolving technological settings. We can identify various generation of works in this area [PAT09]. The first generation of model-based approaches in which the focus was basically on deriving abstractions for graphical UIs (see for example UIDE [FOL94]). At that time, UI designers focused mainly on identifying relevant aspects for this kind of interaction modality. Then, the approaches evolved into a second generation focusing on expressing the high-level semantics of the interaction: this was mainly supported through the use of task models and associated tools, aimed at expressing the activities that the users intend to accomplish while interacting with the application (see for example Adept [JOH93], GTA [DER96], ConcurTaskTrees (CTT) [PAT99]).

Nowadays, the increasing availability of new interaction platforms has raised a new interest in model-based approaches in order to allow developers to define the input and output needs of their applications, vendors to describe the input and output capabilities of their devices, and users to specify their preferences. However, such approaches should still allow designers to have good control on the final result in order to be effective.

4.2.4 The CAMELEON Reference Framework

The CAMELEON Unified Reference Framework [CAL02] [CAL03] was produced by the EU-funded CAMELEON Project [CAM-Proj] and results from two key principles:

- A Model-Based approach
- Coverage of both the design and run-time phases of a multi-target UI

CAMELEON describes a framework that serves as a reference for classifying UIs supporting multiple targets, or multiple contexts of use in the field of context-aware computing. Furthermore, the CAMELEON Framework provides a unified understanding of context-sensitive UIs rather than a prescription of various ways or methods of tackling different steps of development.

4.2.4.1 *The Context of Use*

Context is an all-embracing term. Composed of “con” (with) and “text”, context refers to the meaning that must be inferred from the adjacent text. As a result, to be operational, context can only be defined in relation to a purpose, or finality [CRO02]. In the field of context-aware computing a definition of Context that has been largely used is provided by [DEY00]: *“Context is any information that can be used to characterize the situation of entities (i.e. whether a person, place or object) that are considered relevant to the interaction between a user and an application, including the user and the application themselves. Context is typically the location, identity and state of people, groups and computational and physical objects”*.

While the above definition is rather general, thus encompassing many aspects, it is not directly operational. Hence, we hereby define the **Context of Use** of an interactive system as a *dynamic, structured information space* that includes the following entities:

- a model of the **User**, U, (who is intended to use or is actually using the system)
- the hardware-software **Platform**, P, (which includes the set of computing, sensing, communication, and interaction resources that bind together the physical environment with the digital world)
- the social and physical **Environment**, E, (where the interaction is actually taking place).

Thus, a context of use is a triple composed by **(U, P, E)**

The *User* represents the human being (or a human stereotype) who is interacting with the system. The characteristics modelled or relevant can be very dependant on the application domain. Specific examples are age, level of experience, the permissions, preferences, tastes, abilities and disabilities, short term interests, long term interests, etc. In particular, perceptual, cognitive and action disabilities may be expressed in order to choose the best modalities for the rendering and manipulation of the interactive system.

The *Platform* is modelled in terms of resources, which in turn, determine the way information is computed, transmitted, rendered, and manipulated by users. Examples of resources include memory size, network bandwidth, and input and output interaction devices. [CAL02] distinguishes between *elementary platforms* (e.g. laptop, PDA, mobile phone), which are built from *core resources* (e.g. memory, display, processor) and *extension resources* (e.g. external displays, sensors, mice), and clusters, which are built from elementary platforms. Resources motivate the choice for a set of input and output modalities and, for each modality, the amount of information made available. W3C's Delivery Context Ontology [DCONTOLOGY] is intended to define a concrete Platform Model.

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 behaviour, either now or in the future (Coutaz and Rey, 2002). According to our definition, 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, level of stress...

The relationship between a UI and its contexts of use leads to the following definitions:

Multi-target (or multi-context) UI

A *multi-target (or multi-context) UI* supports multiple types of users, platforms and environments. *Multi-user, multi-platform* and *multi-environment* UIs are specific classes of multi-target UIs which are, respectively, sensitive to user, platform and environment variations. [CAL03]

Adaptive UI

An *Adaptive* UI refers to a UI capable of being aware of the context of use and to (*automatically*) react to changes of this context in a continuous way (for instance, by changing the UI presentation, contents, navigation or even behaviour).

Adaptable UI

An *Adaptable UI* can be tailored according to a set of predefined options. Adaptability normally requires an explicit human intervention. We can find examples of UI adaptability on those word processors where the set of buttons contained by toolbars can be customized by end users.

Plastic UI

A *Plastic UI* is a multi-target UI that *preserves usability* across multiple targets. Usability is not intrinsic to a system. Usability can only be validated against a set of properties set up in the early phases of the development process. [CAL03]

4.2.4.2 Abstraction Levels

The CAMELEON Reference Framework, structures the development life cycle into four levels of abstraction, from the task specification to the running interface (see Figure 2):

- The Task and Concepts level (corresponding to the Computational-Independent Model–CIM–in MDE) which considers: (a) the logical activities (tasks) that need to be performed in order to reach the users' goals and (b) the domain objects manipulated by these tasks. Often tasks are represented hierarchically along with indications of the temporal relations among them and their associated attributes.
- The Abstract User Interface (AUI) (corresponding to the Platform-Independent Model–PIM– in MDE) is an expression of the UI in terms of interaction spaces (or presentation units), independently of which interactor are available and even independently of the modality of interaction (graphical, vocal, haptic …). An interaction space is a grouping unit that supports the execution of a set of logically connected tasks.
- The Concrete User Interface (CUI) (corresponding to the Platform-Specific Model–PSM– in MDE) is an expression of the UI in terms of "concrete interactor", that depend on the type of platform and media available and has a number of attributes that define more concretely how it should be perceived by the user. "Concrete interactor" are, in fact, an abstraction of actual UI components generally included in toolkits.
- The Final User Interface (FUI) (corresponding to the code level in MDE) consists of source code, in any programming language or mark-up language (e.g. Java, HTML5, VoiceXML, X+V, ...). It can then be

interpreted or compiled. A given piece of code will not always be rendered on the same manner depending on the software environment (virtual machine, browser …). For this reason, CAMELEON considers two sublevels of the FUI: the source code and the running interface

These levels are structured with a relationship of reification going from an abstract level to a concrete one and a relationship of abstraction going from a concrete level to an abstract one. There can be also a relationship of translation between models at the same level of abstraction, but conceived for different *contexts of use*. These relationships are depicted on Figure 2.

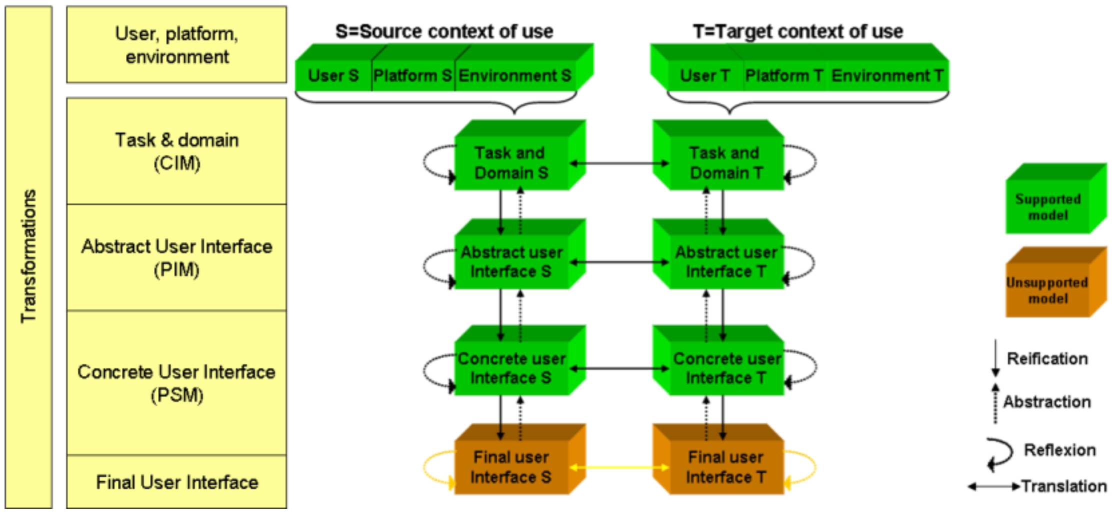


Figure 2 Relationships between components in the CAMELEON Reference Framework

4.2.5 Context of Use Model

Figure 3 depicts graphically the "Context of Use" Model proposed. Such a Model captures the Context of Use in which a user is interacting with a particular computing platform in a given physical environment in order to achieve an interactive task.

Context of Use is the main entity which has been modelled as an aggregation of *User*, *Platform* and *Environment*. They are all *Context Elements*. A *Context Element* is an instance of *Context Entity*. A *Context Property* represents a characteristic of a *Context Element* or information about its state. A *Context Property* might be associated to zero or more instances of *Context Value*. Examples of *Context Property* instances are: 'position', 'age' or 'cpuSpeed'. There can be Context Property instances composed by other sub-properties. For example, the 'position' property is typically composed by: 'latitude', 'longitude' and 'altitude'. Both a *Context Property* and a *Context Value* can be associated to different metadata represented by the *Context Property Description* and *Context Value Description* classes respectively. A *Context Property* can be obtained from different *Context Providers*. A *Device Description Repository* (DDR) [DDR-REQUIREMENTS] [DD-LANDSCAPE] is a *Context Provider* of information about the "a priori known" characteristics of *Platform Components*, particularly devices or web browsers.

The model below also describes a simple, abstract conceptual model for the *Platform*. A *Platform* can be represented as an aggregation of different *Aspect* [DDR-SIMPLE-API] instances (device, web browser, network, camera, ...), which are called *Components*. To this aim we have splitted this relationship into three different aggregations: *active*, *default* and *available*. *active* indicates that a *Component* is "running". For example, if a camera is "on" such a *Component* is said to be "active". *default* conveys what is the referred *Aspect* instance when there is no an explicit mention of a specific one. Finally, *available* represents what are the "ready to run" *Components*. For example, when a device has more than one web browser installed, the "available web browsers" property value will be a list containing a reference to the web browsers that could (potentially) be put into running.

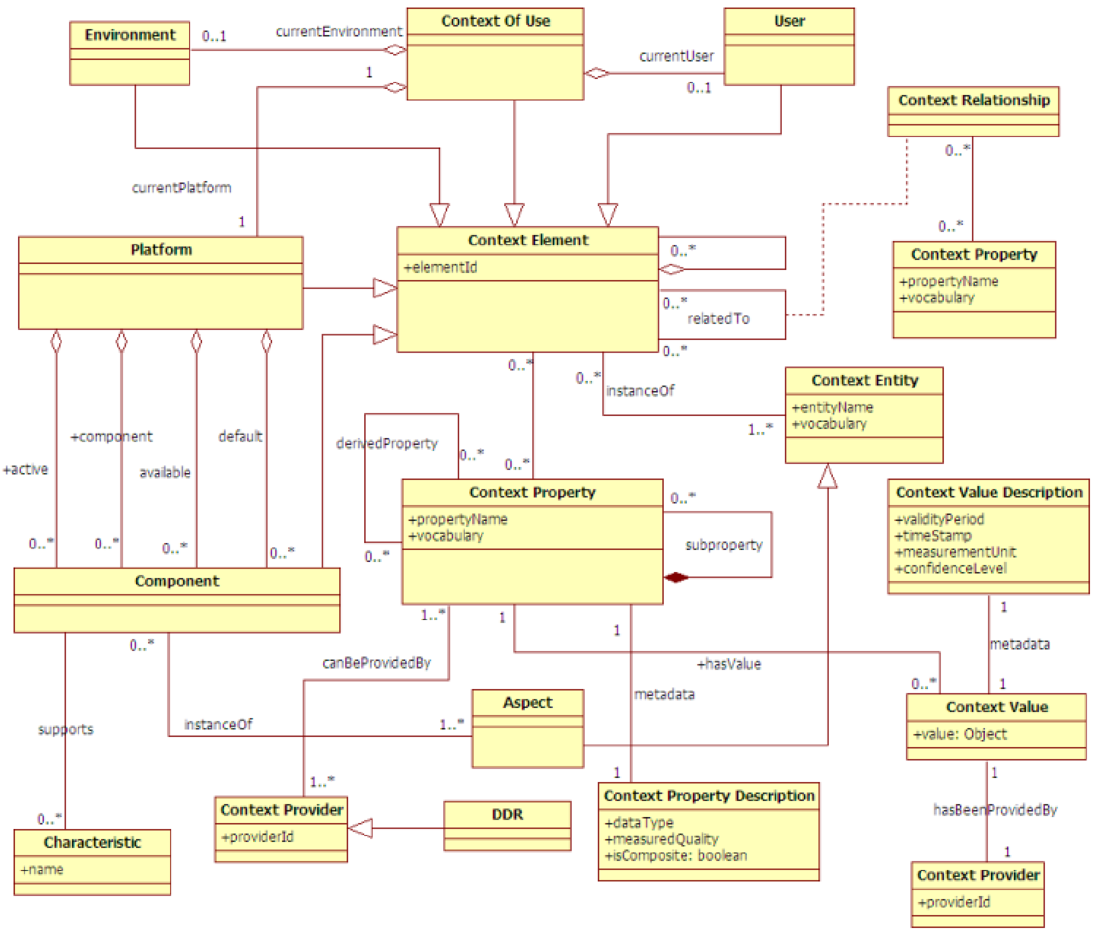


Figure 3 Context of Use Model

4.2.6 UsiXML AUI Meta-Model

The figure below depicts the current version of the UsiXML Meta-Model for AUI description (work in progress). The class *AUIobject* is at the top of the hierarchy representing the elements that populate an AUI Model.

AUIInteractor and *AUIContainer* are subsumed by *AUIObject*. The latter defines a group of tasks that have to be presented together and may contain both *AuiInteractors* or other *AuiContainers*.

An association class *AuiRelationship* allows defining the kind of relationship (*Ordering, Hierarchy, Grouping,* or *Repetition*) between an object and its container. *AUIInteractionUnit* is an aggregation of *AUIObject* and *Behaviour* specified by means of *Listener*, *Event* and *Action*. *AuiInteractor* has been splitted into *DataInteractor* (for UI data input/output) or *TriggerInteractor* (for UI command).

Selection, Input and Output are data interactor. Concerning trigger interactor, *Command* is intended to launch any kind of action within the UI whilst *Navigator* allows to change the interaction unit.

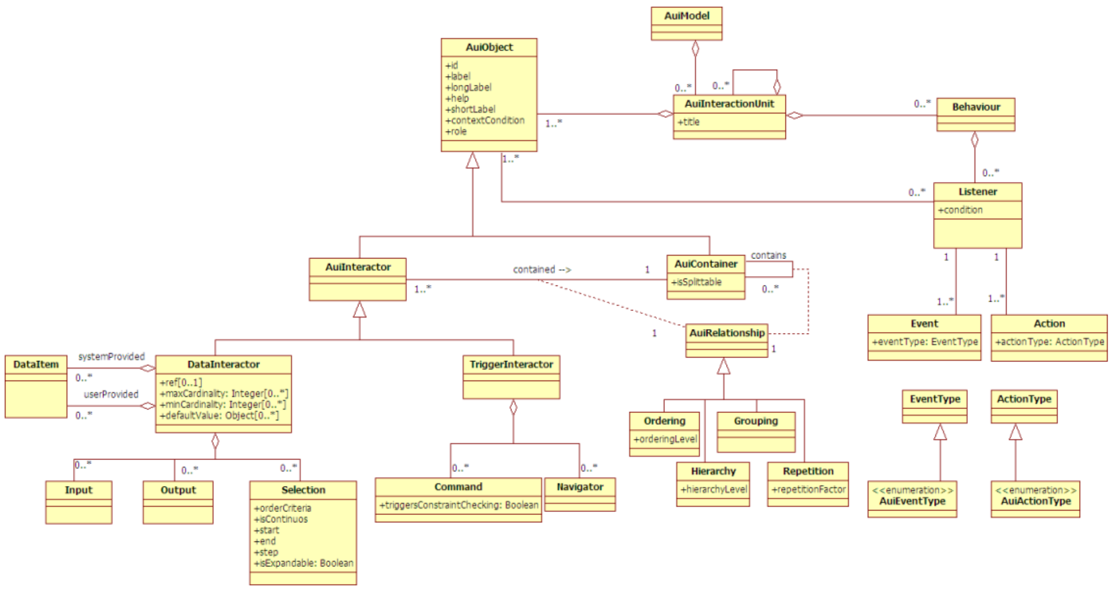


Figure 4 Abstract User Interface Meta-Model in UsiXML

4.2.7 UsiXML CUI Meta-Model

Figure 5 is the graphical representation of the UsiXML Meta-model for the Concrete UI (work in progress). The root entity is *CUIObject* which has been subclassed in *CUIInteractor* and *CUIContainer*. The relationship between Interactor and Containers is captured by the 'contains' relationship and the *CUIRelationship* association class. It is important to note that the meta-model includes specializations for the different modalities (graphical, tactile, vocal), as a CUI Model is modality-dependent. The *Style* class is intended to capture all the presentational attributes for a CUI Object. This design pattern decouples the model from 'presentational vocabularies'.

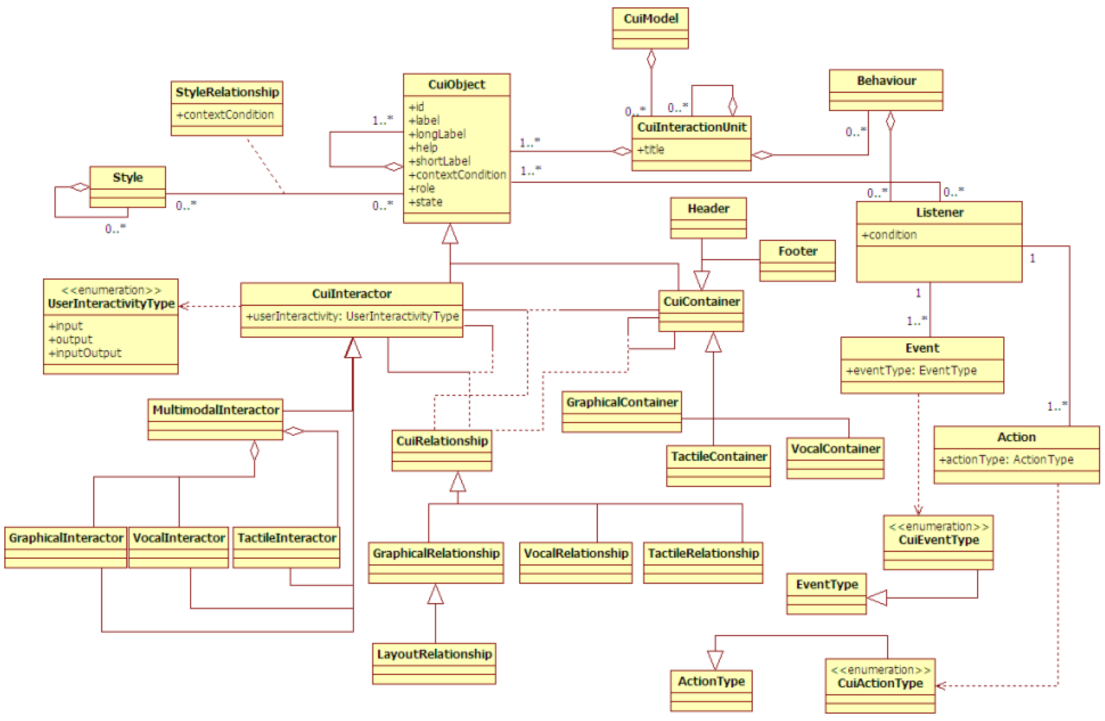


Figure 5 Concrete User Interface Meta-Model in UsiXML

4.3 Contribution to the Reference Specification

4.3.1 Introduction

A detailed chapter on the contribution to the Reference Specification is included in the document D1.1c Appendix A: Contribution to other WPs. In that document are included the high level requirements identified for the presentation concern of NEXOF-RA (service front-end). It includes the identified use cases to put together the main concepts, a description of system requirements and functionalities that should offered by NEXOF-RA.

WP1 has also contributed to the specification with two patterns (Front-End in E-SOA and Mashup as a Service), produced as an outcome of the investigation teams promoted by WP1.

A vision of the Front-End in E-SOA pattern is included in section 4.4. The whole pattern is included in D1.1c Appendix A Section 3.1 Front-End in E-SOA Pattern.

A vision of the Mashup as a Service pattern is included in section 4.5. The whole pattern is included in D1.1c Appendix A Section 3.2 Mashup as a Service Pattern.

Table 4: Patterns produced by each IT

Pattern	WP10 System Requirements	Investigation Team
Front-End in E-SOA	SR 7. How can users be supported to access all the SOA-based system functionalities (Services and Processes creation, management, execution, etc.)?	Declarative UI Authoring Languages
Mash up as a service (MaaS)	SR 1.4. How can a service be executed (run-time support)? SR 5.1. How can processes be designed in terms of the services they are composed of (Orchestration, Choreography descriptions)? SR 5.1.2. How can process be designed in order to manage interoperability matters (data mapping, message transformation)?	Declarative UI Authoring Languages
Context Management		Context Model and Universal APIs

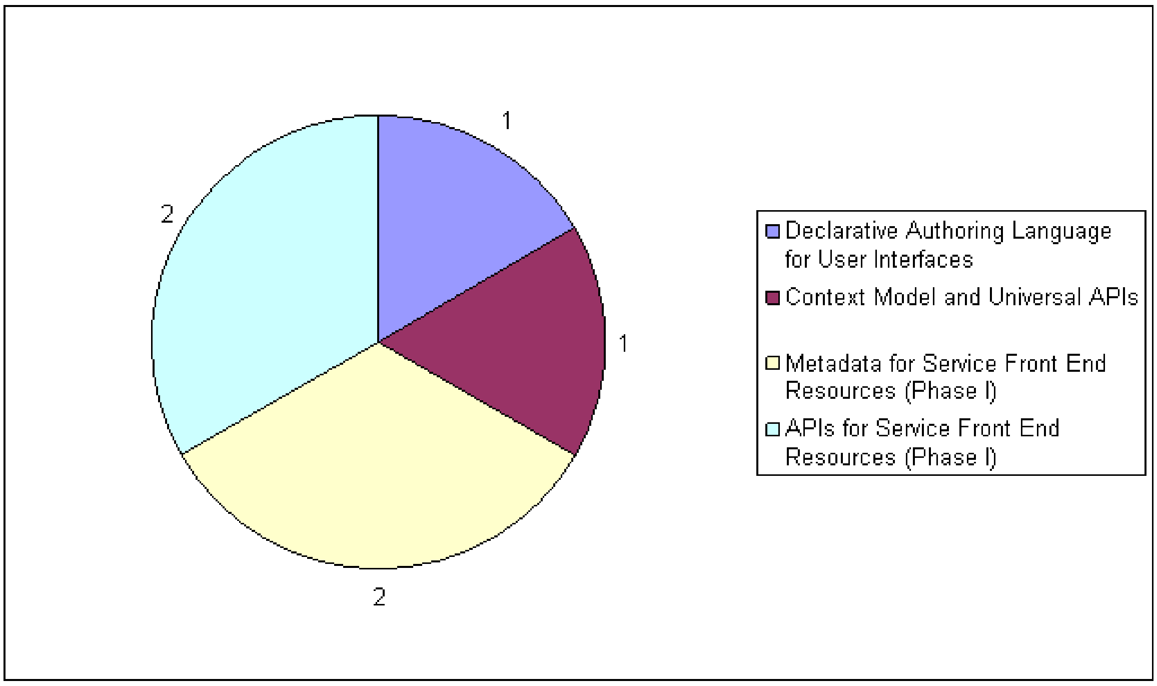


Figure 6 IT and patterns

4.4 Front-End in E-SOA

4.4.1 Abstract

This pattern is a refinement of the “Designer and Runtime Tools for E-SOA” one. Concretely, it focuses on the functionalities provided by software components that are responsible for the design and execution of artefacts devoted to the NEXOF presentation (front-end) concern All the architectural choices made by this pattern are related to the decomposition into three functional aspects: accessing to and adapting SFERs, composing SFERs and engineering SFERs. In addition, the “Context of Use Manager” component, covers the cross-cutting aspect that has to do with the provision of contextual information (what is the device, browser, network, location, etc.) needed to adapt automatically the interaction to different computing platforms, users or environments.

The whole pattern is included in D1.1c Appendix A Section 3.1 Front-End in E-SOA Pattern.

4.4.2 Requirements

The following use-case diagram shows the functional requirements that this pattern meets.

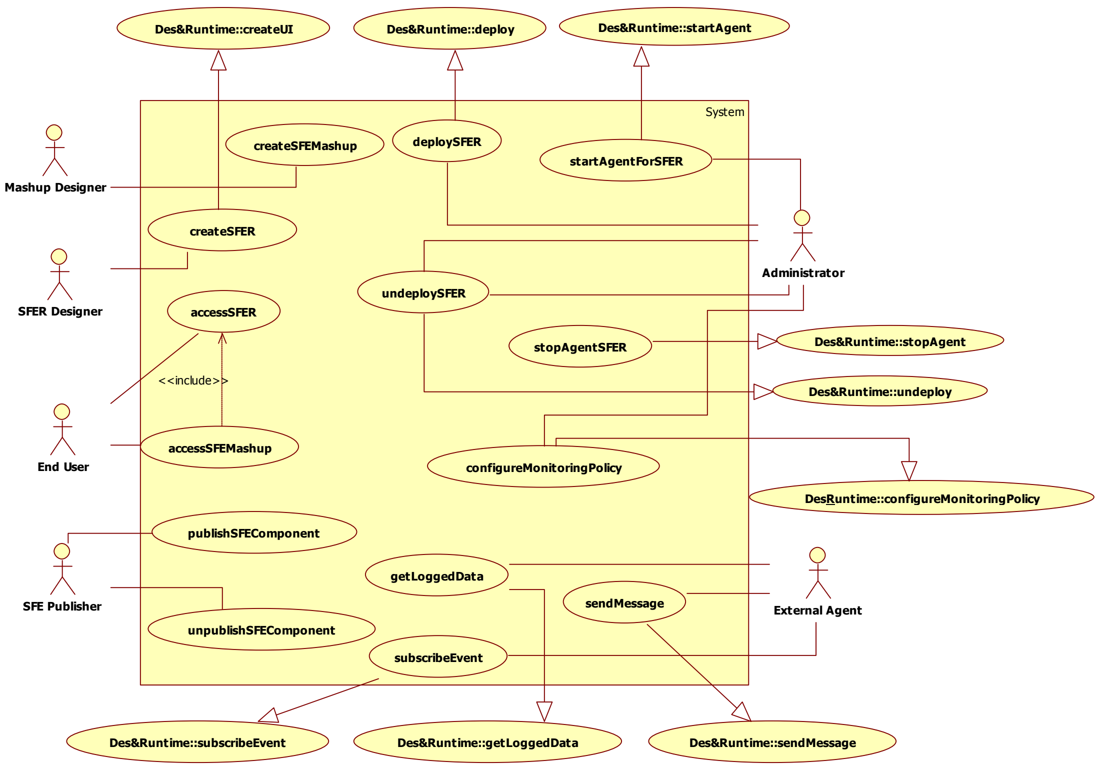


Figure 7 Functionalities provided by the Front End in ESOA

- **createSFER**: this functionality enables the creation of SFE Resources. It is a refinement of the generic functionality **createUIComponent** provided by the Designer and Runtime Tools (DRT) pattern.
- **accessSFER**: this functionality allows users to interact with SFE Resources. This is a specific functionality introduced by E-SOA Front-End pattern and hence it is not related to any functionality of the DRT pattern.
- **deploySFER** allocates computational resources for the SFER which refines the **deploy** functionality provided by the DRT pattern. For example, consider a SFE Resource which it is a Java-based Web Application. The **deploySFER** functionality would be achieved by copying a WAR file and the addition of a new application context in the container configuration files.
- **startAgentForSFER** starts an agent that realizes the SFER which refines the **startAgent** functionality offered by the DRT pattern. In fact this functionality puts into execution a SFE Resource. Once this operation has finished the SFE Resource is ready to be accessed.

  For example, consider a SFE Resource which it is a Java Servlet-based web application. Provided that such a SFE Resource has been formerly deployed, the **startAgentForSFER** functionality would be achieved by starting the container (Tomcat) (if not formerly running) and enabling the corresponding web application.

- **undeploySFER** frees the computational resources of the NCI needed for its execution. This functionality refines the **undeploy** functionality provided by the DRT pattern. In the Java-based Web Application example this operation would correspond to the deletion of the war file.
- **stopAgentSFER** that stops the corresponding agent that realizes the SFER. It refines **stopAgent** from the DRT pattern. In the Java-based Web Application example this operation would correspond to the disablement of the application context within the Java container.
- **createSFEMashup**: this functionality makes it possible the creation of SFE Mashups. It includes two other functionalities (**createWorkspace** and **composeSFERs**) which are part of the solution described on Section 5. This is an specific functionality introduced by this pattern and hence it is not related to any functionality of the DRT pattern.
- **accessSFEMashup**: this functionality allows users to interact with SFE Mashups . It includes two other functionalities (**selectWorkspace** and **wireSFERs**) which are part of the solution described on Section 5. This is an specific functionality introduced by this pattern and hence it is not related to any functionality of the DRT pattern.
- **publishSFEComponent**: this functionality allows a user to publish a SFE Component in a SFE Catalogue. From the point of view of this pattern, publishing a SFE Component means that the SFE Component is registered in the SFE Catalogue, and hence it is ready to be used by users of the SFE Mashup Platform. This is an specific functionality introduced by this pattern and hence it is not related to any functionality of the DRT pattern.
- **unpublishSFEComponent**: this functionality allows a user to unpublish a SFE Component in a SFE Catalogue. From the point of view of this pattern, unpublishing a SFE Component means that the SFE Component is unregistered in the SFE Catalogue, and hence it is no longer available to be used by users of the SFE Mashup Platform. This is an specific functionality introduced by this pattern and hence it is not related to any functionality of the DRT pattern.
- **getLoggedData**: this functionality corresponds to the generic functionality of the same name described by the DRT pattern, applied to the specific case of Service Front-Ends.
- **sendMessage**: this functionality corresponds to the generic functionality of the same name described by the DRT pattern, applied to the specific case of Service Front-Ends.
- **subscribeEvent**: this functionality corresponds to the generic functionality of the same name described by the DRT pattern, applied to the specific case of Service Front-Ends.

4.4.3 Applicability

The "Front-End in E-SOA" pattern enables the development of user interfaces capable to adapt to different computing platforms, users or environments, thus "**Adaptation to new operating environments**" is positively affected. Additionally, the **usability** of the system is increased as the adaptation processes will be aimed to provide a harmonized user experience across multiple devices or modes of interaction.

Nonetheless, the front-end mash-up approach has security issues, particularly with respect to the **isolation** of components. As the different SFERs within a mash-up are typically executed under the same client platform (for example a web browser), sensitive information can be exposed between SFERs or denial-of-service attacks can be performed by malign SFERs.

Additionally, the proposed pattern (if mash-up is used) impacts negatively on the "**resource efficiency**". Indeed, each SFER is not aware of the operative status of the other SFERs and this can jeopardize the application of ad hoc resource allocation policies that allows the optimization of the resources usage.

Final, the architectural choice that separates the mash-up layer from SFER access layer improves the **availability** of the system, as SFERs can be accessed even if mash-ups are not available for any reason.

4.5 Mashup as a Service

4.5.1 Abstract

This pattern is part of the Internet of Services pattern, it provides an architectural solution for a subset of components belonging to the IoS pattern. In particular, this pattern provides the architecture of the components belonging to User Interface and Service Execution concerns of the IoS.

All the architectural choices made by this pattern are related to the decomposition into four functional aspects: Designing the Mashup, Accessing and using the Mashup, Delivery of a workspace and storing and managing of Meta-Information:

- the "*Mashup Designer"* This component enables a Mashup Creator to design a Mashup.
- The "*Mashup Access*" enables a Mashup User to access and use a Mashup.
- The "*Workspace Manager*" enables the creation, edition, configuration and delivery of a workspace and all its content: (Mashup, gadget, etc…)
- The "*Mashup Metadata*" is in charge of store all the Meta information about the mashups, allowing the publication, tagging and searching of mashups and its individual elements.

The whole pattern is included in D1.1c Appendix A Section 3.2 Mashup as a Service Pattern.

4.5.2 Requirements

The following use-case diagram shows the functional requirements that this pattern meets.

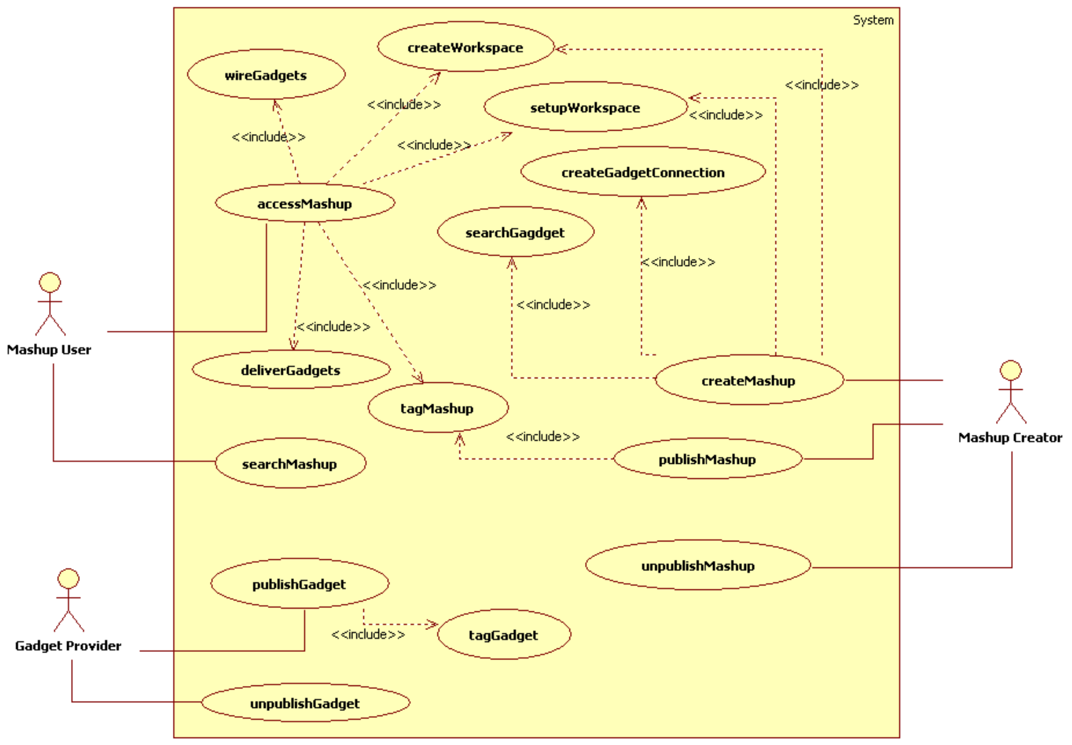


Figure 8 Functionalities provided by the MaaS Pattern

- **createWorkspace**: this functionality enables the creation of an empty Workspace ready to be used by a Mashup Creator or a Mushup User
- **setupWorkspace**: this functionality is in charge of the configuration the workspace including the insertion of new gadget and determining its size and position.
- **createMashup**: this functionality allows a Mashup Creator to compose a new Mashup, including the searching of the appropriate gadgets (*searchGadget* functionality) insertion the gadget in the workspace, determining its size and position (*setupWorkspace* functionality), the creation of new channels and creating the connection between the gadgets (*createGadgetConnection* functionality).
- **publishMashup**: this functionality allows a Mashup Creator to publish the selected Mashup in the catalogue, including the meta information about the mashup, its gadgets and the connection between the gadgets. A published mashup can be searched by a Mashup User, to use it.
- **unpublishMashup**: this functionality allows a Mashup Creator to unpublish from the catalogue a published Mashup.
- **publishGadget**: this functionality allows a Gadget Provider to publish the selected Gadget in the catalogue, including the meta information about

the gadget. It also includes the functionality to tag the gadget with the selected labels. A published gadget can be searched by a Mashup Creator, to create a Mashup.

- **unpublishGadget**: this functionality allows a Gadget Provider to unpublish from the catalogue a published Gadget.
- **createGadgetConnection**: defines a connection between two different gadgets, for interchanging information. One of the gadgets acts as a source of information and provides an input for the second gadget.
- **tagMashup**: this functionality allows a Mashup Creator and a Mashup User to catalogue a mashup with the selected tag. Associating a tag to a mashup makes it easier to be found.
- **tagGadget**: this functionality allows a Gadget Provider to catalogue a gadget with the selected tag. Associating a tag to a gadget makes it easier to be found.
- **searchMashup**: this functionality allows a Mashup User to make a search in a catalogue looking for the desired mashup.
- **searchGadget**: this functionality allows a Mashup Creator to make a search in a catalogue looking for the desired gadget.
- **accessMashup**: this functionality allows a Mashup User to access a previously created mashup. This functionality deals with the security constraints and the delivery and wiring of the gadgets that are in the mashup.
- **deliverGadget**: this functionality allocates the resources needed by the gadget, making the deploy of the gadget in the workspace.
- **wireGadget**: this functionality is in charge of implementing the interchange of information from the different gadgets of a mashup, as is defined in the connections of the gadgets.

4.5.3 Applicability

The proposed pattern aims to improve the **integrability** of service front-ends. Indeed, front-end mash-ups are an enabler to make separately services developed as Gadgets work correctly together. This pattern also allows connecting and composing services improving the “**Extension of Capacity**” and **reusability**.

The “Mashup as a Service” pattern enables the development of user interfaces that can be executed in different Mashup Platforms available in different environments, thus “**Adaptation to new operating environments**” is positively affected.

Nonetheless, this pattern approach has security issues, particularly with respect to the **isolation** of components. As the different Gadgets within a mash-up are typically executed under the same client platform, sensitive information can be exposed between Gadgets or denial-of-service attacks can be performed by malign Gadgets.

Additionally, the proposed pattern impacts negatively on the "**resource efficiency**". Indeed, each Gadget is executed without being aware of the others Gadget execution status. This could be a risk for the application of ad hoc resource allocation policies that allows the optimization of the resources usage.

Final, the architectural choice that integrates the Gadgets in a Mashup Platforms, decrease the **availability** of the system, as it is combination of components provided by different providers that needs to be all available at the same time.

4.6 Context of Use Management

4.6.1 Abstract

The presented pattern defines the architecture of a software infrastructure for managing the *Context of Use* of an interactive system. The proposed architecture identifies three different layers. The *API layer* (provision and monitoring), which offers a programmatic interface for getting access and for monitoring the Context of Use. The *mediation layer*, in charge of providing a uniform view of the Context of Use, hiding the complexities of gathering (possibly distributed) contextual information. And finally, the *sensing layer* which knows all the technical details (including protocols) needed to gather the information from the different Context Providers.

The whole pattern is included in D1.1c Appendix A Section 3.3 Context of Use Management Pattern.

4.6.2 Requirements

Here after a subset of the conceptual model is reported. The defined concepts are necessary for the description of the functional requirements. These are here reported for completeness reasons.

The figure below depicts graphically the "Context of Use" Model proposed by the NEXOF-RA Project. Such a Model captures the Context of Use in which a user is interacting with a particular computing platform in a given physical environment in order to achieve an interactive task.

Context of Use is the main entity which has been modelled as an aggregation of *User*, *Platform* and *Environment*. They are all *Context Elements*. A *Context Element* is an instance of *Context Entity*. A *Context Property* represents a characteristic of a *Context Element* or information about its state. A *Context Property* might be associated to zero or more instances of *Context Value*. Examples of *Context Property* instances are: 'position', 'age' or 'cpuSpeed'. There can be *Context Property* instances composed by other sub-properties. For example, the 'position' property is typically composed by: 'latitude', 'longitude' and 'altitude'. Both a *Context Property* and a *Context Value* can be associated to different metadata represented by the *Context Property Description* and *Context Value Description* classes respectively. A *Context Property* can be obtained from different Context Providers. A Device Description Repository (DDR) is a *Context Provider* of information about the "a priori known" characteristics of Platform Components, particularly devices or web browsers.

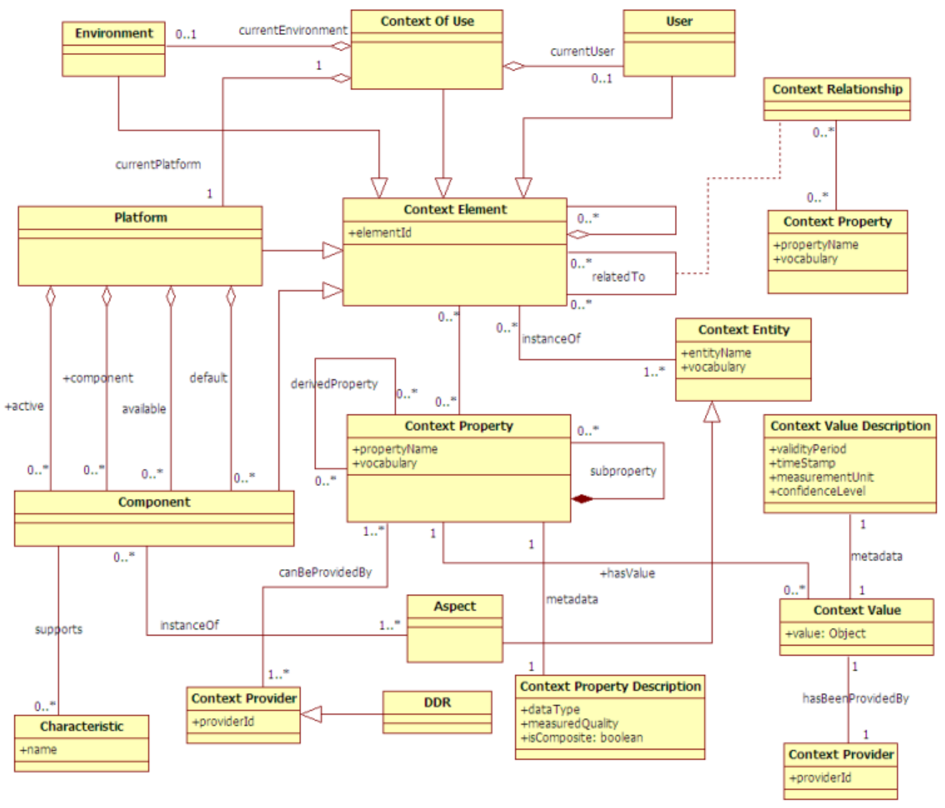


Figure 9 Functionalities provided by the Context Of Use Management

The following use-case diagram shows the functional requirements that this pattern meets.

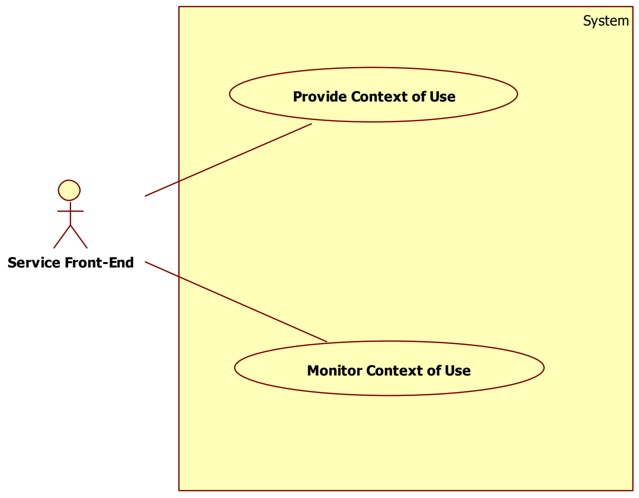


Figure 10 Functionalities provided by the Context Of Use Management

- **<u>provideContextOfUse</u>**: this functionality enables the retrieval of the values of the context properties of the different context elements from a set of context providers available in the system.
- **<u>monitorContextOfUse</u>**: this functionality enables the monitoring of the values of the context properties of the different context elements from a set of context providers available in the system.

4.6.3 Applicability

The proposed pattern affects positively to the **integrability**, as different and heterogeneous Context Providers can work within the system, provided a connector is made available to the sensing layer. For the same reasons the pattern aims to increase the **extension of capability**.

The pattern aims to increase the **reusability** by defining a mediation layer which it is independent of the context provision infrastructure. In addition the pattern is independent of the Context Properties and it can work with any property that fits in the proposed Context Model.

The pattern promotes an authorization schema that controls every access to a Context Property. Hence **security, privacy** and **auditability** are improved.

Finally, it has to be remarked that the **response time** is negatively affected due to both the granularity of the security checks and the division of the system in three layers which adds a significant overhead. Particularly, the dynamic invocations that are issued by the sensing layer to the different Context Providers.

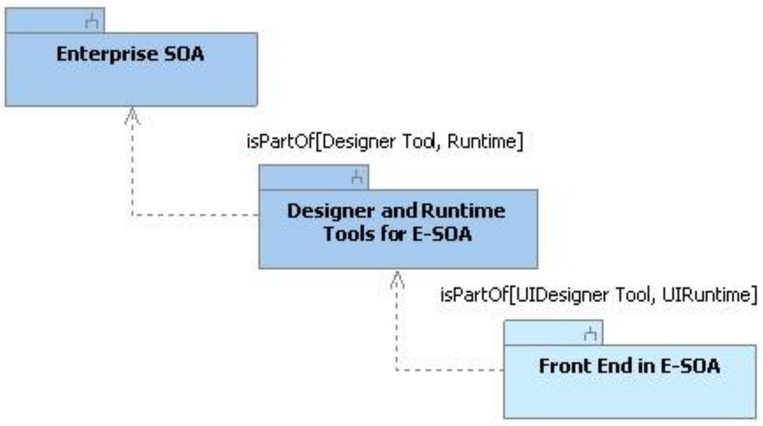
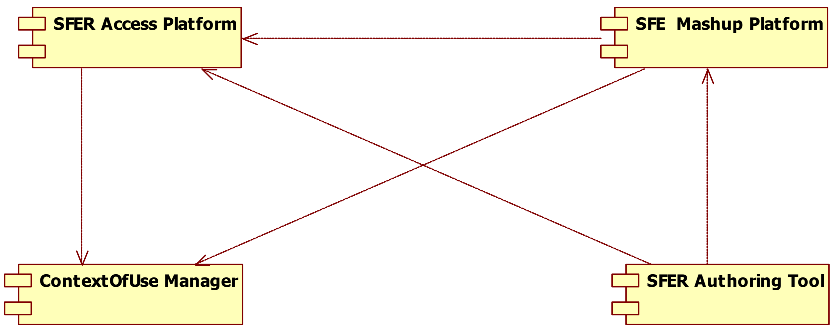
5 Contribution to the PoCs

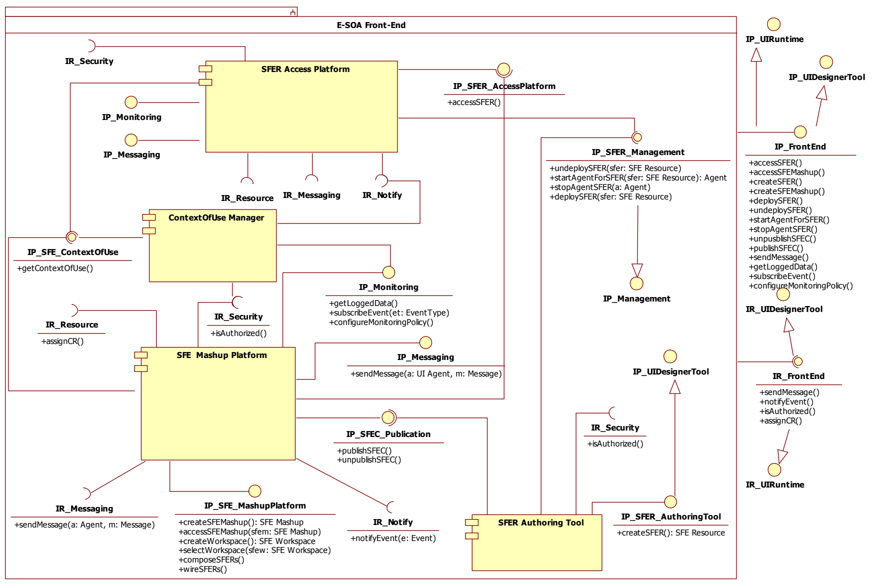
5.1 PoC Front End E-SOA

5.1.1 PoC Description

POC Name	**Develop a next generation service front-end (SFE) application for Business systems**
What	
Owners	TIE/TID This PoC has been selected and setup based on a proposal coming from WP1 (TID) and took over by TIE based on work performed and achieved by WP1 in close cooperation with WP7.
Description of the PoC	This PoC will be used to experiment with the main guiding principles and architectural concepts for user-service interaction, as proposed by NEXOF-RA.
ACPs involved	Essentially in order to fulfill declared principles this abstract pattern introduces the following components and relationships among them: • SFER Authoring Tool • SFE Mashup Platform • SFER Access Platform • ContextOfUse Manager On the communication level EzWeb abstract pattern makes use of several possible alternatives for enforcing dynamic communication between humans and services on the web: • Generation of (X)HTML fragments that could be incorporated into a master page • Generation of data that should be interpreted by client application (browser), e.g. JSON and YAML
Objective of the PoC	Service Oriented Architecture (SOA) has attracted a great deal of interest during the last years. In fact SOA increase asset reuse, reduce integration expenses and improve businesses agility in responding to new demands. Nonetheless, until now, the mainstream development and research around SOA has been focused mainly on middleware and scalability, service engineering and automating service composition using Business Process Management (BPM) technologies. Little or no attention has been put on common/standardized service front-ends which are, in our vision, necessary to provide a complete SOA and services framework. As a consequence SOA remain on a technical layer hidden to the end-user. The evolution of web-based interfaces is a testimony of the progress achieved to improve service usability. However, existing, web-based service front-ends are far from completely meeting end-user expectations. Applications and information portals still are based on monolithic, inflexible, non-context-aware, and non-customizable. As a consequence end-users do not really benefit from the advantages promoted by service orientation in terms of modularity, flexibility and composition. In addition service front-ends are constructed in an ad-hoc manner and with the absence of formal engineering methods and tools

	that could accelerate the time to market. To resolve these shortcomings of current SOA approaches NEXOF-RA advocates a new generation Service Front layer that will allow flexible creation of applications by the end users themselves. The basic requirements that are addressed by the NEXOF-RA approach, and that will be put to test in this PoC, are: • UI_ACC: How SOA-based systems can be accessed by users? • UI_COMP: How users can compose their own, personalized interface? • UI_COMP_DES: How users can design their own operating environments? • UI_COMP_FIND: How users can find service front end resources? • UI_COMP_CON: How users can connect service front end resources?
Functionalities	The PoC focus on the functionalities located at the “Presentation” key concern. The Front-End in E-SOA pattern is part of the Designer and Runtime Tools for E-SOA pattern. This pattern provides the architecture of the UI Runtime and the UI Designer Tool. More specifically the UIRuntime has been refined into three main components: the SFER Access Platform, the SFE Mashup Platform and the Context of Use Manager. At the present level of abstraction the UI Designer Tool has only been refined into the SFE Authoring Tool.
Dependencies	The PoC is related to the following concepts in the Service Front-End (SFE) layer architecture: • Service Front-End Resources and Workspaces • SFE Resources Catalogues • Service-Front Resources integration with back-end services and systems • End User empowerment The following diagram shows the relationship of the “Front-End in E-SOA” Pattern with other patterns.

	 Enterprise SOA isPartOf[Designer Tool, Runtime] Designer and Runtime Tools for E-SOA isPartOf[UIDesigner Tool, UIRuntime] Front End in E-SOA Dependencies with other ESOA patterns  SFER Access Platform SFE Mashup Platform ContextOfUse Manager SFER Authoring Tool Components of the SOA Front-End pattern and their dependencies The Front-End in E-SOA pattern is part of the Designer and Runtime Tools for E-SOA pattern. This pattern provides the architecture of the UI Runtime and the UI Designer Tool. More specifically the UIRuntime has been refined into three main components: the SFER Access Platform, the SFE Mashup Platform and the Context of Use Manager. At the present level of abstraction the UI Designer Tool has only been refined into the SFE Authoring Tool. An specific pattern will give more details about such component.
Why	
Rationale	Proposed as a phase I PoC, it will focus on the main architectural components and prepare the way for more complex, joint PoCs and, where applicable, validation of specifications. It illustrates how different Service Front-end resources can be combined by the end users to address a particular problem and proves the benefits of this approach versus the traditional static applications. By doing so it also demonstrates the suitability of some of the main elements of the proposed architecture (workspaces, catalogues, loose integration between front-end and back-end resources based on web services and REST approaches etc.).
Architecture Component(s)	• The concept of Service Front End Resource implementing the user-service interaction

affected	• The concept of "Workspace Editing Tool" in which users edit their own workspace by composing and connecting different Service Front End Resources • The concept of Workspace Access Layer which it is the runtime support that allows users to access their previously composed workspaces • The concept of Resource Catalogue and how users can upload new Service Front End Resources to their workspaces
Alternatives	The alternatives in this case are limited to machine specific APIs such as SOAP or RESTFul.
Relationship with the NEXOF-RA	Proposed abstract pattern relies on the following required functionalities from NEXOF-RA key concerns: messaging, security, virtualization and monitoring.
How	
Scenarios for validation	The PoC involves developing and deploying of several gadgets. Some of them are related to the existing services from real-world applications. Two of them taken from a TIE's pool of services. They are developed with .NET, they are a part of large scale industrial application of TIE. Another one is from an external party, namely Google Maps. They are exposed to EzWeb as gadgets and then an end user is able to create a mash-up or composition of these gadgets via GUI, saves his workspace with services and share new composite service with other users.
Suggested Architecture	The PoC will be based on the platform developed by the EzWeb project. This project has been the source of contributions for the service front-end layer in the NEXOF-RA architecture. It is provided as open source software and already implements part of the target architecture.  E-SOA Front-End IR_Security SFER Access Platform IP_Monitoring IP_Messaging IR_Resource IR_Messaging IR_Notify IP_SFER_AccessPlatform +accessSFER() IP_SFER_Management +undeploySFER(sfer: SFE Resource) +startAgentForSFER(sfer: SFE Resource): Agent +stopAgentSFER(a: Agent) +deploySFER(sfer: SFE Resource) IP_Management IP_UIRuntime IP_UIDesignerTool IP_FrontEnd +accessSFER() +accessSFEMashup() +createSFER() +createSFEMashup() +deploySFER() +undeploySFER() +startAgentForSFER() +stopAgentSFER() +unpusblishSFEC() +publishSFEC() +sendMessage() +getLoggedData() +subscribeEvent() +configureMonitoringPolicy() IR_UIDesignerTool IR_FrontEnd +sendMessage() +notifyEvent() +isAuthorized() +assignCR() IR_UIRuntime ContextOfUse Manager IP_SFE_ContextOfUse +getContextOfUse() IP_Monitoring +getLoggedData() +subscribeEvent(et: EventType) +configureMonitoringPolicy() IR_Resource +assignCR() IR_Security +isAuthorized() SFE Mashup Platform IP_Messaging +sendMessage(a: UI Agent, m: Message) IP_UIDesignerTool IR_Security +isAuthorized() IP_SFEC_Publication +publishSFEC() +unpublishSFEC() IR_Messaging +sendMessage(a: Agent, m: Message) IP_SFE_MashupPlatform +createSFEMashup(): SFE Mashup +accessSFEMashup(sfem: SFE Mashup) +createWorkspace(): SFE Workspace +selectWorkspace(sfew: SFE Workspace) +composeSFERs() +wireSFERs() IR_Notify +notifyEvent(e: Event) SFER Authoring Tool IP_SFER_AuthoringTool +createSFER(): SFE Resource An architecture of front-end for SOA pattern As part of this specific scenario used to validate the concepts, the following service front-end resources have to be developed • SFER 1 List of existing partners. A secured component that shows

	the list of existing partners for user • SFER 2 Partner data. A secured component that shows detailed information of a partner • SFER 3 Map. A generic SFER map that shows a location on a map “ala Google Maps” These SFER resources will be published by means of the EzWeb catalogue and they will be used to build new composite resources through the workspace and wiring functionalities provided by the EzWeb platform.
Environment	Please, follow the installation instructions specific for the software listed below. • EzWeb platform [http://demo.ezweb.morfeo-project.org/] • OS: Windows • Apache 2.2 [apache_2.2.11-win32-x86-no_ssl.msi] • PostgreSQL 8.3 [postgresql-8.3.1-1.zip] • Python 2.5 [python-2.5.4.msi] • Django 1.0 [Django-1.0.2-final.tar.gz] • libxml2dom-0.4.7 [libxml2dom-0.4.7.tar.gz] • libxml2-python-2.7.3 [libxml2-python-2.7.3.win32-py2.5.exe] • mod_python-3.3.1 [mod_python-3.3.1.win32-py2.5-Apache2.2.exe] • PIL-1.1.6 [PIL-1.1.6.win32-py2.5.exe] • psycopg2-2.0.6 [psycopg2-2.0.6.win32-py2.5-pg8.2.4-release.exe] • PyXML-0.8.4 [PyXML-0.8.4.win32-py2.5.exe] • Google maps [http://maps.google.com] • TIE authentication service [proprietary implementation in .Net]
Methods	The method used to validate architectural choices was prototyping with parallel expert estimations of the given statements. More specifically methods for usability, availability and inerrability are elaborated separately in the following section.
Assessment criteria, metrics, and a way to document	To validate the stated pattern qualities (non-functional properties) the following assessment criteria and metrics should be used. There are a number of *usability* evaluation methods (UEMs) to assess and improve usability of systems, which defines assessment criteria and metrics for them. It is still an ongoing discussion in academic circles concerning the comparison and effectiveness of these methods. Since usability study is out of the scope of NEXOF-RA so just several informal interviews were conducted in order to evaluate how a produced prototype meets listed user requirements and address architectural needs for SOA presentation concern. For *availability* assessment criterion a metric formulates as a ratio of the expected value of the uptime of a system to the aggregate of the expected values of up and down time. Another dimension of *availability* for front-end systems is the possibility to access them by challenged people, people with limited connectivity or from old software/hardware systems. This can be evaluated against specific metrics for each case, e.g.

	• Is it enough contrast? *(type I)* • Is font big enough? *(type I)* • Does browser support JavaScript for running client side logic? *(type II)* Type I questions should be also classified as a sort of usability properties. As such they should be developed and investigated further with the methods listed above. In general, those questions will be answered using prototyping and user studies. Type II falls into a category of an expert-based assessments. In other words, experienced software architect, designer, developer or groups of those creates a list of possible options and then gives answers on them based on a provided architecture.
Further Information	A demo shows applicability of architectural and design principals proposed by WP1 on a base of previously isolated services (web applications).

The Future Internet will give an active role to end-users letting them author, enhance, share and customize their own applications and operating environment (workspace), thus going beyond traditional, monolithic user-service interaction

5.1.2 Scope of the PoC with respect the Reference Architecture

- Contextual scenario:
  - e-Commerce (S12)
- Addressed requirement:
  - UI_ACC: How SOA-based systems can be accessed by users?
  - UI_COMP: How users can compose their own, personalized interface ?
  - UI_COMP_DES: How users can design their own operating environments?
  - UI_COMP_FIND: How users can find service front end resources?
  - UI_COMP_CON: How users can connect service front end resources?
- Patterns proven:
  - Interaction and Architecture Patterns to guarantee a certain high level of usability of a service oriented ecosystem for an end user [Service Front-End Resources and Workspaces, Service Front-End Resources Catalogues, Service Front-End Resources integration with back-end services and systems, End User empowerment interaction and architectural patterns]
- Proof of concept environment:
  - TIE services, EzWEb framework by TID and third party services (e.g., Google maps)

5.1.3 Summary of the results and conclusions

QA of a set of patterns (9) have been evaluated leading to the identification of tradeoffs and sensitivity points.

Results of PoCs Phase I enable architects using NEXOF-RA to perform architectural choices based on tradeoffs between the different alternatives

- Proves that different Service Front-end resources can be combined by the end users to address a particular problem
- Proves the benefits of this approach versus the traditional static applications
- More and more requested by service consumers and consequently from service providers

6 APPENDIX A: CONTRIBUTION TO OTHER WPS

See the document D1.1c Appendix A: Contribution to other WPs

7 APPENDIX B: INVESTIGATION TEAMS CONTRIBUTIONS

See the document D1.1c Appendix B: Investigation team contributions

REFERENCES

[CAL02]

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

[CAL03]

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

[CAM-Proj]

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

[CAN09]

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

[CRO02]

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

[DCONTOLOGY]

José Manuel Cantera Fonseca; Rhys Lewis. *Delivery Context Ontology.* 16 June 2009. W3C Working Draft. (Work in progress.) URL: http://www.w3.org/TR/2009/WD-dcontology-20090616/

[DER96]

VAN DERVEER,G., LENTING, B., ANDBERGEVOET,B. 1996. *GTA: Groupware task analysis—Modelling complexity*. Acta Psychologica 91, 297–322.

[DEY00]

A. K. Dey. *Providing architectural support for building context-aware applications*. PhD thesis, Georgia Institute of Technology, Atlanta, GA, USA, 2000. Director-Gregory D. Abowd.

[DEY01]

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

[FOL94]

Foley, D. and Noi Sukaviriya. *History, results, and bibliography of the user interface design environment (UIDE), an early model-based system for user interface design and implementation*. In Proceedings of Design, Verification and Specification of Interactive Systems (DSVIS'94). 3–14. 1994

[FLO06]

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

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

[JOH93]

JOHNSON, P., WILSON, S., MARKOPOULOS, P., AND PYCOCK, J. *ADEPT: Advanced design environment for prototyping with task models. In Proceedings of the International Conference on Human Computer Interaction and ACM Conference on Human Aspects on Computer Systems (INTERCHI). 56. 1993*

[MariaXML]

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

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

[MBUI]

A W3C Workshop on "Future Standards for MBUI" has been celebrated on 13-14 May 2010 in Rome. The final report of the XG is at http://www.w3.org/2005/Incubator/model-based-ui/XGR-mbui-20100504/

[OPEN]

OPEN (Open Pervasive Environments for migratory iNteractive services) (See http://www.ict-open.eu)

[PAT03]

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

[PAT05]

Paternò F., Model-based Tools for Pervasive Usability, Interacting with Computers, Elsevier, May 2005, Vol.17, Issue 3, pp. 291-315.

[PAT09]

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

[PAT99]

F.Paternò, *Model-based Design and Evaluation of Interactive Applications*, Springer Verlag, November 1999, ISBN 1-85233-155-0

[RAG09]

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

[SEF04]

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

[SFE OA]

Open Alliance on Services Front Ends.

(See http://sfe.morfeo-project.org/)

[USIXML]

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

[W3C04]

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

[W3C08]

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

[W3C09]

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

[W3C07]

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