



D2.1

Use case definition and requirements

Version 4.0

WP 2: Requirements, business models and definition of the architecture

Dissemination Level: Public

Lead Editor: TID

19/05/2008

Status: Final version

SIXTH FRAMEWORK PROGRAMME

PRIORITY IST-2005/2006-2.5.5



Service and Software Architectures, Infrastructures and Engineering

Proposal/Contract no.: 034101

*Mr. Alberto M. León Martín
Telefónica I+D
OPUCE Coordinator*

This is a public deliverable that is provided to the community under the license Attribution-NoDerivs 2.5 defined by creative commons <http://www.creativecommons.org>

This license allows you to

- to copy, distribute, display, and perform the work
- to make commercial use of the work

Under the following conditions:



Attribution. You must attribute the work by indicating that this work originated from the IST-OPUCE project and has been partially funded by the European Commission under contract number IST-034101



No Derivative Works. You may not alter, transform, or build upon this work without explicit permission of the consortium

- For any reuse or distribution, you must make clear to others the license terms of this work.
- Any of these conditions can be waived if you get permission from the copyright holder.

This is a human-readable summary of the Legal Code below:

License

THE WORK (AS DEFINED BELOW) IS PROVIDED UNDER THE TERMS OF THIS CREATIVE COMMONS PUBLIC LICENSE ("CCPL" OR "LICENSE"). THE WORK IS PROTECTED BY COPYRIGHT AND/OR OTHER APPLICABLE LAW. ANY USE OF THE WORK OTHER THAN AS AUTHORIZED UNDER THIS LICENSE OR COPYRIGHT LAW IS PROHIBITED.

BY EXERCISING ANY RIGHTS TO THE WORK PROVIDED HERE, YOU ACCEPT AND AGREE TO BE BOUND BY THE TERMS OF THIS LICENSE. THE LICENSOR GRANTS YOU THE RIGHTS CONTAINED HERE IN CONSIDERATION OF YOUR ACCEPTANCE OF SUCH TERMS AND CONDITIONS.

1. Definitions

- "Collective Work"** means a work, such as a periodical issue, anthology or encyclopedia, in which the Work in its entirety in unmodified form, along with a number of other contributions, constituting separate and independent works in themselves, are assembled into a collective whole. A work that constitutes a Collective Work will not be considered a Derivative Work (as defined below) for the purposes of this License.
- "Derivative Work"** means a work based upon the Work or upon the Work and other pre-existing works, such as a translation, musical arrangement, dramatization, fictionalization, motion picture version, sound recording, art reproduction, abridgment, condensation, or any other form in which the Work may be recast, transformed, or adapted, except that a work that constitutes a Collective Work will not be considered a Derivative Work for the purpose of this License. For the avoidance of doubt, where the Work is a musical composition or sound recording, the synchronization of the Work in timed-relation with a moving image ("synching") will be considered a Derivative Work for the purpose of this License.
- "Licensor"** means all partners of the OPUCE consortium that have participated in the production of this text
- "Original Author"** means the individual or entity who created the Work.
- "Work"** means the copyrightable work of authorship offered under the terms of this License.
- "You"** means an individual or entity exercising rights under this License who has not previously violated the terms of this License with respect to the Work, or who has received express permission from the Licensor to exercise rights under this License despite a previous violation.

2. Fair Use Rights. Nothing in this license is intended to reduce, limit, or restrict any rights arising from fair use, first sale or other limitations on the exclusive rights of the copyright owner under copyright law or other applicable laws.

3. License Grant. Subject to the terms and conditions of this License, Licensor hereby grants You a worldwide, royalty-free, non-exclusive, perpetual (for the duration of the applicable copyright) license to exercise the rights in the Work as stated below:

- a. to reproduce the Work, to incorporate the Work into one or more Collective Works, and to reproduce the Work as incorporated in the Collective Works;
- b. to distribute copies or phonorecords of, display publicly, perform publicly, and perform publicly by means of a digital audio transmission the Work including as incorporated in Collective Works.
- c. For the avoidance of doubt, where the work is a musical composition:
 - i. **Performance Royalties Under Blanket Licenses.** Licensor waives the exclusive right to collect, whether individually or via a performance rights society (e.g. ASCAP, BMI, SESAC), royalties for the public performance or public digital performance (e.g. webcast) of the Work.
 - ii. **Mechanical Rights and Statutory Royalties.** Licensor waives the exclusive right to collect, whether individually or via a music rights society or designated agent (e.g. Harry Fox Agency), royalties for any phonorecord You create from the Work ("cover version") and distribute, subject to the compulsory license created by 17 USC Section 115 of the US Copyright Act (or the equivalent in other jurisdictions).
- d. **Webcasting Rights and Statutory Royalties.** For the avoidance of doubt, where the Work is a sound recording, Licensor waives the exclusive right to collect, whether individually or via a performance-rights society (e.g. SoundExchange), royalties for the public digital performance (e.g. webcast) of the Work, subject to the compulsory license created by 17 USC Section 114 of the US Copyright Act (or the equivalent in other jurisdictions).

The above rights may be exercised in all media and formats whether now known or hereafter devised. The above rights include the right to make such modifications as are technically necessary to exercise the rights in other media and formats, but otherwise you have no rights to make Derivative Works. All rights not expressly granted by Licensor are hereby reserved.

4. Restrictions. The license granted in Section 3 above is expressly made subject to and limited by the following restrictions:

- a. You may distribute, publicly display, publicly perform, or publicly digitally perform the Work only under the terms of this License, and You must include a copy of, or the Uniform Resource Identifier for, this License with every copy or phonorecord of the Work You distribute, publicly display, publicly perform, or publicly digitally perform. You may not offer or impose any terms on the Work that alter or restrict the terms of this License or the recipients' exercise of the rights granted hereunder. You may not sublicense the Work. You must keep intact all notices that refer to this License and to the disclaimer of warranties. You may not distribute, publicly display, publicly perform, or publicly digitally perform the Work with any technological measures that control access or use of the Work in a manner inconsistent with the terms of this License Agreement. The above applies to the Work as incorporated in a Collective Work, but this does not require the Collective Work apart from the Work itself to be made subject to the terms of this License. If You create a Collective Work, upon notice from any Licensor You must, to the extent practicable, remove from the Collective Work any credit as required by clause 4(b), as requested.
- b. If you distribute, publicly display, publicly perform, or publicly digitally perform the Work or Collective Works, You must keep intact all copyright notices for the Work and provide, reasonable to the medium or means You are utilizing: (i) the name of the Original Author (or pseudonym, if applicable) if supplied, and/or (ii) if the Original Author and/or Licensor designate another party or parties (e.g. a sponsor institute, publishing entity, journal) for attribution in Licensor's copyright notice, terms of service or by other reasonable means, the name of such party or parties; the title of the Work if supplied; and to the extent reasonably practicable, the Uniform Resource Identifier, if any, that Licensor specifies to be associated with the Work, unless such URI does not refer to the copyright notice or licensing information for the Work. Such credit may be implemented in any reasonable manner; provided, however, that in the case of a Collective Work, at a minimum such credit will appear where any other comparable authorship credit appears and in a manner at least as prominent as such other comparable authorship credit.

5. Representations, Warranties and Disclaimer

UNLESS OTHERWISE MUTUALLY AGREED TO BY THE PARTIES IN WRITING, LICENSOR OFFERS THE WORK AS-IS AND MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE MATERIALS, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, WARRANTIES OF TITLE, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR THE ABSENCE OF LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OF ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO SUCH EXCLUSION MAY NOT APPLY TO YOU.

6. Limitation on Liability. EXCEPT TO THE EXTENT REQUIRED BY APPLICABLE LAW, IN NO EVENT WILL LICENSOR BE LIABLE TO YOU ON ANY LEGAL THEORY FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES ARISING OUT OF THIS LICENSE OR THE USE OF THE WORK, EVEN IF LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. Termination

- a. This License and the rights granted hereunder will terminate automatically upon any breach by You of the terms of this License. Individuals or entities who have received Collective Works from You under this License, however, will not have their licenses terminated provided such individuals or entities remain in full compliance with those licenses. Sections 1, 2, 5, 6, 7, and 8 will survive any termination of this License.

- b. Subject to the above terms and conditions, the license granted here is perpetual (for the duration of the applicable copyright in the Work). Notwithstanding the above, Licensor reserves the right to release the Work under different license terms or to stop distributing the Work at any time; provided, however that any such election will not serve to withdraw this License (or any other license that has been, or is required to be, granted under the terms of this License), and this License will continue in full force and effect unless terminated as stated above.

8. Miscellaneous

- a. Each time You distribute or publicly digitally perform the Work, the Licensor offers to the recipient a license to the Work on the same terms and conditions as the license granted to You under this License.
- b. If any provision of this License is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this License, and without further action by the parties to this agreement, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.
- c. No term or provision of this License shall be deemed waived and no breach consented to unless such waiver or consent shall be in writing and signed by the party to be charged with such waiver or consent.
- d. This License constitutes the entire agreement between the parties with respect to the Work licensed here. There are no understandings, agreements or representations with respect to the Work not specified here. Licensor shall not be bound by any additional provisions that may appear in any communication from You. This License may not be modified without the mutual written agreement of the Licensor and You.

Context

WP 2	<i>Requirements, business models and definition of the architecture</i> This Work Package aims to analyse the use cases and scenarios where OPUCE platform can be applied, as well as the business vision that guides the OPUCE platform, by providing business use cases involving different stakeholders. Once a representative number of scenarios will be proposed, a deep analysis of needs and requirements at all the levels will be the starting point for the future design and development of the whole infrastructure, their components and interfaces.
WPL	TID
Task 2.1	<i>Use case definition and specification of requirements</i> This task deals with the identification of the most promising scenarios and ambiances, including on-demand adaptive services and end-users creations, as well as the definition of requirements to guide the technical modelling of OPUCE components and platform. It will guarantee coherence, interoperability and future harmonious evolution of the designed platform.
TL	TID
Dependencies	This deliverable will be the basis for D2.3 and D2.4.
Starting date	26/11/2007
Release date	19/05/2008

Author(s)	TID
Contributor(s)	UVA; ERI; HUA; RHA; NEC; PTI; TIL; UPM
Reviewers	ALU; DAV; PTO
Approved by:	IPCC

Table of Contents

- 1. Introduction..... 15
- 2. OPUCE definitions..... 17
 - 2.1. Base service 17
 - 2.2. Service..... 17
 - 2.3. Service Description..... 18
 - 2.3.1. Service Logic facet 18
 - 2.4. Semantic Description facet..... 18
 - 2.5. Service package 18
 - 2.6. Platform modules..... 18
 - 2.6.1. User Information Management module 19
 - 2.6.2. Portal module 19
 - 2.6.3. Service Advertising module 19
 - 2.6.4. Service Execution module 20
 - 2.6.5. Context Awareness module..... 20
 - 2.6.6. Service Lifecycle Management module 20
 - 2.7. Scenario 20
 - 2.8. Use case..... 21
 - 2.9. Actors 21
 - 2.10. Users and roles (of OPUCE platform) 21
 - 2.11. Verification and validation..... 22
 - 2.12. Service life cycle 23
 - 2.13. Service life cycle management 23
 - 2.14. Service Creation Environment 23
 - 2.15. Service Editor 23
 - 2.16. Service Deployment..... 24
 - 2.17. Service Undeployment..... 24
 - 2.18. Service Provisioning 24
 - 2.19. Service Creation/Composition 24
 - 2.20. Service Simulation 24
 - 2.21. Direct Service Advertising..... 25
 - 2.22. Service Sharing 25
 - 2.23. Service Recommendation..... 25
 - 2.24. Personalization 25
 - 2.25. Service Adaptation..... 26

3.	OPUCE user profile and context	27
3.1.	OPUCE Users profiling.....	27
3.1.1.	OPUCE service producers and consumers.....	27
3.1.2.	Base service producers.....	30
3.2.	Terminals.....	31
3.2.1.	Service development.....	32
3.2.2.	Service usage.....	33
3.3.	User context	33
3.3.1.	User Context in OPUCE	35
4.	Use Case description	36
4.1.	An OPUCE story.....	36
4.2.	OPUCE platform elements	37
4.3.	OPUCE service consumer Use Cases	38
4.3.1.	Service look-up.....	38
4.3.2.	Service subscription, personalization and usage	40
4.3.3.	Direct Service Advertising	41
4.3.4.	Service recommendation.....	42
4.3.5.	Remote Service Management Use Case	44
4.3.6.	Context Usage and Feed Use Case.....	45
4.4.	OPUCE service producer Use Cases.....	46
4.4.1.	Service composition	47
4.4.2.	Service management.....	49
4.4.3.	Service Simulation.....	51
4.4.4.	Service Sharing	53
4.5.	Base service producer Use Cases	54
4.5.1.	Base Service creation.....	54
4.5.2.	Base Service management	56
5.	Requirements	58
5.1.	Guidelines.....	58
5.2.	Service look-up requirements.....	58
5.3.	Service Subscription, Personalization and Usage requirements	58
5.4.	Direct Service Advertising Requirements	60
5.5.	Service Recommendation Requirements	60
5.6.	Remote Service Management requirements	61
5.7.	Context Usage and Feed requirements.....	62
5.8.	Service Composition Requirements	63
5.9.	Service management requirements.....	64

5.10.	Service Simulation requirements	65
5.11.	Service Sharing requirements	66
5.12.	Base Service Creation requirements	66
5.13.	Base Service Management requirements.....	68
5.14.	Platform modules classification of requirements	69
6.	Conclusions.....	80
Annex A.	Examples of OPUCE services	81
A.1.	Auto-conference service	81
A.2.	Advanced Reachability service.....	82
A.3.	Zaragoza International Exposition 2008	82
A.4.	Meeting Agreement	83
A.5.	Massive On-line Multiplayer Game Clan Community	83
A.6.	Recipe on-line ordering	84
A.7.	Location Based Photo Sharing Album – Mobisaic.....	85
A.8.	The Big Thinker	86

List of Figures

Figure 1. OPUCE platform users and roles.....22

Figure 2. Overall percentage usage of tools/services across age.....28

Figure 3. Ratios of contribution to viewing for groups of services29

List of Tables

Table 1. OPUCE Service Consumers and Service Producers per age.....30

Table 2. Worldwide: Preliminary PDA Vendor Shipment Estimates by Operating System. ...32

Table 3. Categorization of environmental context.....34

Table 4. Requirements classification by platform modules and end user device.....79

Abbreviations

AAA	Authentication, Authorization and Accounting
ACL	Access Control List
Aml	Ambient Intelligence
CE	Component Editor
CER	Context Enabler and Reasoner
CUF	Context Usage and Feed
ED	End Device
GSD	Graphic Service Descriptor
GSM	Global System for Mobile Communications
GUI	Graphic User Interface
HLE	History and Learning Engine
ICT	Information and Communications Technology
IDE	Integrated Development Environment
IM	Instant Messaging
IMS	IP Multimedia Subsystems
IPR	Intellectual Property Rights
IT	Information Technology
MB	Matcher Broker
MP	Matcher Provider
MPCC	Multi Party Call Control
OMA	Open Mobile Alliance
OPUCE	Open Platform for User-centric service Creation and Execution
OS	Operating System
P2P	Peer to Peer
PC	Personal Computer
PDA	Personal Digital Assistant
PoC	Push to Talk Over Cellular
Qos	Quality of Service
RFC	Request For Comment
RSM	Remote Service Management
SCE	Service Creation Environment
SEE	Service Execution Environment
SIP	Session Initiation Protocol
SLA	Service Level Agreement
SME	Small and Medium Enterprise
SMS	Short Message Service

SPML	Services Provisioning Markup Language
TBT	The Big Thinker and Family
UI	User Interface
UIM	User Information Management
UMTS	Universal Mobile Telecommunications System
UPR	User Profile Repository
WP	Work Package

Executive Summary

The OPUCE project will produce an open service infrastructure to enable users for easy service creation and deployment in heterogeneous environments and ambiances, allowing services to be accessed in a seamless way by a multitude of devices connected via different networks. It will have tangible benefits to end-users as it will enrich services variety and relevance to their needs, and to service providers as it will ease the service creation and extend the scope of services that can be offered. It will leverage the business of Small to Medium Enterprises (SMEs) by facilitating quick creation and deployment of a new range of services as “smart users” by ad-hoc cooperation with other business players. From this, the number of service providers and users will increase and consequently the amount of equipment sold by manufacturers, which will provide the enabling platform elements

The present document comes as a result of the work developed within the framework of the OPUCE project and concretely WP2 aimed at analysing the use cases and scenarios where OPUCE platform can be applied. Specifically, this deliverable is the evolution of first phase D2.1 (DRAFT version) which has been completed during the second phase of the project. D2.1 now presents an updated list of use cases, that represent more accurately the key features and actions to be performed thanks to OPUCE and consequently, an updated list of requirements extracted from these use cases.

As previous D2.1, the objective of the paper is to describe a representative number of use cases from where the requirements required by these use cases will be identified. This is an important task since these requirements identified are the starting point for the design and development of OPUCE components and platform.

Thus, the deliverable is divided into four different parts.

The first part of the document presents an extensive and complete list of definitions related to OPUCE. This section not only comprises terms related to use cases and requirements but all terms that have been used in the whole project. It is aimed at being used as a general guide for the project and it clearly describes the most important concepts and the meaning they have in the OPUCE context.

The second section is all about user profiling and context. This section introduces the types of users that will be more eligible to use the OPUCE platform in order to give an idea of the acceptance the OPUCE platform will have among these users. It also introduces the different contexts in where the platform can be used and also describes the terminal requirements needed to use OPUCE. Since OPUCE main objective is to enable end-users to create and experience their own services, it is important to distinguish among the different types of end users accessing the platform and the ambiances they interact with. These features, such as the terminals used, just to provide an example, will therefore influence in the aspect and usability of the GUI as well as in the underlying architecture.

Once all the previous characteristics are introduced, the third part of the Deliverable provides a set of use cases used to justify the use of the OPUCE platform. The collection of use cases is grouped in three subsets, one for each of the OPUCE main actors: the service consumer, the service producer and the base service producer. In general terms, the collection of use cases shows how the OPUCE platform facilitates the creation of services through the provision of a repository of base services that will be combined in order to produce the intended service; as well as the management of the whole lifecycle of the service (including subscription, deployment, execution, etc), personalization of services and social network facilities (sharing, direct advertising, etc). Also shown here is how the platform offers the possibility to create base services from scratch, in this case just to service providers, not end-users. This way, OPUCE services are presented as a flexible set of heterogeneous components instead of a tightly coupled set of components, as it would be currently.

Finally, the last section of the document presents the definition of requirements. The collection of requirements has been identified from the previously described use cases and they are therefore presented divided by the use case from which they were extracted. The definition of requirements represents one of the main objectives of this task, since it will guide the technical modelling of OPUCE functional elements and platform.

Some of these requirements represent OPUCE innovation challenges and highlight the fact that OPUCE services are envisioned as short-lived services that are adaptable to the environment they are deployed in, i.e. depending on the execution platform and taking into account the end-user surroundings and communication constraints.

1. Introduction

Nowadays a bunch of applications (multimedia, messaging, videoconferencing, presence, infotainment, etc.) have burst in the market with multiple features covering different environments and networks. The operators develop most of these applications, also because they have the user data and the network knowledge. Operators and big companies usually develop solutions with a great potential market, but these services require long deployment times and do not always fit on specific user needs. Sometimes the requirements of the users are changing so rapidly and so far from the operator offers.

In such environment, the necessity of a common platform becomes the basis, since without it, each organization is left to build their own set of proprietary business protocols, methods and services, increasing costs, times and efforts and leaving little flexibility for true services collaboration.

OPUCE is a project aimed at developing a complete and global platform for the dynamic creation and delivery of user services and software. *Services* in OPUCE are envisioned as short-lived services that are adaptable to the environment they are deployed in, i.e. depending on execution platform and taking into account the end-user surroundings and communication constraints. *Services* are considered as a provider-client interaction that provides value. In OPUCE, end-users will create OPUCE services by combining base services.

On the other hand, these *base services* are functional units deployed by the operator or authorised third parties, available at the OPUCE platform which can be used to create a complex OPUCE service. *Base services* are atomic from the point of view of the OPUCE users (although they can be composed of one or more functional units/*components* themselves). Users will be able to create a new OPUCE service through composition of several base services.

Base services can be also considered as service *components* from the point of view of tools or utilities that are inherent part of the OPUCE platform and provide specific aspects for service creation, composition and deployment.

In order to better understand the scope of the OPUCE project, and the objectives it addresses, the best way is to take typical use cases on the different aspects of the service lifecycle. The corresponding sections within this deliverable address these use cases. Use cases describe a task or a series of tasks that users will accomplish using the OPUCE platform, and includes the responses of the platform to user actions. Each use case is aimed at explaining how the system should interact with the users to achieve a specific business goal or function. Use case actors may be either end users or other systems.

The main OPUCE Characteristics that should be addressed by OPUCE use cases are:

- End-user Service Creation: End-users and third party service developers can build and manage innovative and integrated services in an easy and interactive way.
- End-user Service Lifecycle Management, focused also on short-time lived services (e.g. hours). Service life cycle management is the sequence of steps of deployment (including download, installation, configuration, activation, and publishing), update (including download, installation, and activation), and removal/withdrawal of services. In OPUCE it does not include service creation and evolution.
- Adaptability and context awareness: Adaptability of services according to context awareness regarding users, networks and terminals.
- Personalisation: Possibility of customization of pre-existing services and components based on personal needs and user profile.

- Service discovering and notification in a dynamic environment, aware of context changing and cooperating in an intelligent way to meet the end-users' demand and preferences.
- Service sharing (among a defined group of people: Individual, community, global, etc.). Storage on a portal and change of privacy attributes for sharing with a community.
- Secured Service Execution, use of security parameters and features of OPUCE for not harming the platform.

2. OPUCE definitions

2.1. Base service

Base services are functional units created and deployed by the operator or authorised third parties, available at the OPUCE platform which can be combined to create a complex OPUCE service.

Base services are atomic from the point of view of the OPUCE users (although they can be composed of one or more functional units). In this sense a base service provides a kind of functionality, able to receive messages from other base services and/or other module of the OPUCE platform and/or to generate new messages.

A base service might need some external software modules to fulfil its purpose. It is expected that these modules are secure and do not harm the platform. They might only be deployed by the platform operator or trusted third parties.

Base services may be end-user oriented base services (messaging, presence, location, etc.), or non-functional base services used for composing the overall OPUCE service logic (timer, if/else branches, etc).

It is possible for OPUCE users to create a new OPUCE service through composition of these base services with other base services. OPUCE accommodates new base services not yet standardized, upon which complex services might be created and also. Besides, some existing enables can be wrapped as base services to be used in the compositions..As an example, enablers standardized by the OMA (presence, group management, PoC, etc).

2.2. Service

A service is a provider-client interaction that provides value. OPUCE services (or services for short) can be created by end-users by combining base services in a syntactical correct way. Such combination may be sequences of components, conditional branches, forks and join operations, etc.

In the OPUCE context, a service contains the following descriptions:

- the complete set of functionality (the service logic, described using BPEL or any other similar language)
- the (semantic) description of the service to be provided to the users (for notification)
- the provisioning information for the service to be configured and to properly activate all those specific service modules associated to the operators service provisioning (set up accounting information, databases updates, etc.)
- one or more pre-defined service lifecycles, which are customized by service creators to fit with execution requirements (i.e. to be available only for a month or to be used only one time)
- the user personalisation and the user interface(s) (if required)

2.3. Service Description

In OPUCE, services are described using a service description (also referenced as service specification in OPUCE documents) which contains all aspects of a service. Each aspect, called a facet, is further described in a separate XML-based specification. With a view to all the functionalities available in the OPUCE platform we have considered three sets of facets:

- Functional facets, which include service logic facets, service interface facets, service semantic information facet, etc.
- Non-functional facets, which include service level agreement (SLA) facets, group management, etc.
- Management facets, which include service lifecycle schedule and deployment facets.

Some facets will be created by OPUCE platform automatically, while others will be filled in by the end-users based on a template.

The service description is detailed in D3.1 [1]

2.3.1. Service Logic facet

The Service Logic is a sequence of processes/functions used to provide a specific service. It includes what base services are used for, in what order, how they have to be used and how to combine them to provide the complete service.

2.4. Semantic Description facet

The Semantic Description consists of meta-information that describes the service in the terms 'What is it for, What it does', etc. It is useful to search for services or service components, discover new services and classify them.

2.5. Service package

The service package is the combination of all service description facets, base service description facets and base service implementation that are needed to execute the logic of the service. It is the entity deployed and managed by the OPUCE platform, which ties together all the parts needed for the service to be successfully deployed, provisioned executed and billed.

2.6. Platform modules

OPUCE platform modules are hardware, middleware and software that implement the OPUCE platform architecture [2] and their interfaces [3]. High-level platform elements are:

- User Information Management module
- Portal module
- Service Advertising module

- Service Execution module
- Context Awareness module
- Service Lifecycle Management module

2.6.1. User Information Management module

In the OPUCE architecture, User Information Management (UIM) is in charge of the storage, management and analysis of user related data. It uses a secured way to keep the critical and sensitive information about users such as their name, address, present location, service subscriptions, personalization preferences, contact list, dynamic context information and usage history data, etc. Additionally, the user's Authentication, Authorization and Accounting (AAA) information is also stored in this module.

2.6.2. Portal module

The Portal module provides the front end interfaces for the different actors to interact with the OPUCE platform. Those interfaces can be used for user provisioning, service creation and composition service management, service subscription and execution and platform administration. The Portal provides visual interface for Service Creators to create new services, edit existing ones and manage the repository of service components. The Portal also enables Service Administrators (Operators) to configure packaged services and deploy them in the runtime environment. There are several sub-blocks inside the portal:

- System Administration
- Web Editor
- Mobile Editor
- Component Editor
- User portal (User Management, Contact Management, Service Galley, Direct Advertising, Service Management, Service Subscription, Service Notification)

2.6.3. Service Advertising module

The Service Advertising module enables solicitation of services to users via multiple channels. It also allows sharing of services between various levels of system participants – from operators to end users.

Whenever a service is created, it is necessary to advertise platform users in order to let them know about its existence. The service advertising module will allow the creator to select specific target users as well as matching of user interests and service description to determine the users interested in that kind of service.

For sharing, a user is able to specifically notify other users (typically his friends) about interesting service.

For recommendation, a matching engine performs correlation of user categories to service features, the Service Advertising module performs correlations and collaborative filtering techniques, which aim to improve end user experience through recommendation of interesting services.

2.6.4. Service Execution module

The OPUCE Service Execution Environment module is responsible to execute OPUCE services. Besides fulfilling performance, scalability and reliability usual requirements, it also aims at unifying heterogeneous execution environments, defining interfaces for external elements that provide functionalities in OPUCE service compositions, the Base Services.

2.6.5. Context Awareness module

The Context Awareness module is in charge, mainly, of performing context aware implicit adaptation, in which the adaptation is performed automatically by the platform (implicitly), without the need of neither creating the service using context aware base service nor “drawing” the service flow using context data explicitly.

Besides this implicit adaptation, the Context Awareness modules is in charge of performing other types of adaptations, more “esthetical”, such as adapting the User Interface to the user’s device characteristics so the UI would be presented correctly in a PC, PDA, mobile phone and so.

2.6.6. Service Lifecycle Management module

The OPUCE platform allows users to create, manage and use their own telecom-based services in an Internet style. One of the key concepts of OPUCE is the complete management of user generated services in its complete life-cycle, this requires in addition to service creation and service execution modules some platform middleware, which supports the transitions between the different phases of the service. Such transitions include the deployment of service logic from an abstract service creation description into the service execution environment, the activation of the services according to scheduled actions, the monitoring of the service usage and the execution states and the undeployment of the services at the end of their life-cycle. The Service Lifecycle Management module aggregates all platform modules involved in those functionalities, including service storage.

2.7. Scenario

A scenario is a narrative describing foreseeable interactions of types of users (characters) and the system or between two software components. Scenarios are often used in usability research. They include information about goals, expectations, actions and reactions. Scenarios are neither predictions nor forecasts. In usability research scenarios are used to describe the situation in which a certain appliance can be used. It is a practical description, written in plain language without any technical detail, which allows non-expert users to understand them and confirm if they describe a real situation. Scenarios are often used to describe the task to the user that is going to perform the test.

2.8. Use case

The definition of use cases is a technique for capturing functional requirements of systems. Each use case provides one or more scenarios that convey how the system should interact with the users called actors to achieve a specific business goal or function. Use case actors may be end users or other systems. Use cases describe a task or a series of tasks that users will accomplish using the OPUCE platform, and includes the responses of the platform to user actions. Use cases are totally independent of the potential services or base services involved in the scenarios.

Present document shows a series of these use cases, that are aimed at extracting the user requirements for the platform. Deliverable D2.3 [2] contains a similar list with names which sometimes are equal to those in this list of requirements, but it is important to point out that the ones showed in D2.3 are system workflows and not use cases. These system workflows are related to the architecture modules and not user requirements. Both concepts appearing in D2.1 and D2.3 have the same name in many cases, which may lead to confusion, so this clarification is needed, since they represent different ideas.

2.9. Actors

In OPUCE platform, actors involved can be:

- **Operator:** a network or telecom operator, or IT organisation.
- **Service Provider:** whoever that is able to provide services to end-users (i.e. enterprises, corporate use, SMEs or even most advanced end-users)
- **End User:** a physical person or individual
- **Software house:** an IT company specialized in producing software components and modules
- **Platform manufacturer/technology provider:** company specialized in building telco and/or IT infrastructures and platforms upon which OPUCE platform is build

Such actors may play different roles according to the ones defined in the next subsection. Typically, the Operator plays the OPUCE Platform owner/administrator role, the software house acts as third party provider/base service producer, and the platform manufacturers play the role of 'platform providers'. End users play different roles as described next.

2.10. Users and roles (of OPUCE platform)

In OPUCE, user is any entity that uses the platform to achieve a specific goal. Depending on the goal, we distinguish between the following roles:

- **Producers:** These users create new services that could be accessed through the OPUCE platform. As we have distinguished between OPUCE services and base services, we need also to distinguish between OPUCE service producers and base service producers.
 - **Base service producers.** This set of users will use the OPUCE platform to create new base services. These new base services will be used to compose new OPUCE services. Base service production will require some expertise in the use of ICT technology, and thus this role is thought to be played by some **third party providers** (software house).

- **OPUCE service producers.** This set of users will use the OPUCE platform to create new services by the composition of the base services already available. OPUCE service productions will not require technical skills and thus this role is thought to be played by non-expert end-users, such as telco customers.
- **Consumers:** These users consume OPUCE services that are offered by the OPUCE platform. OPUCE is aimed at non-skilled users, thus the role of service consumer will be mainly played by non-expert end-users.
- **Administrator:** The Administrator is the actor involved in making everything from the platform side works properly. They are involved also, in the economic flows between the different members (creators, feature creator, etc.).
- **Platform provider:** The Platform Provider provides the platform in which a variety of services run. It is the owner of the platform.

It is possible that sometimes the same user plays two roles: users can use services created by someone else and then they are playing a consumer role, but they can also create their own services that better fit their needs and in this case they are playing an OPUCE service producer role. In such a case we can use the Web 2.0 **prosumer** term to refer to this twofold role.

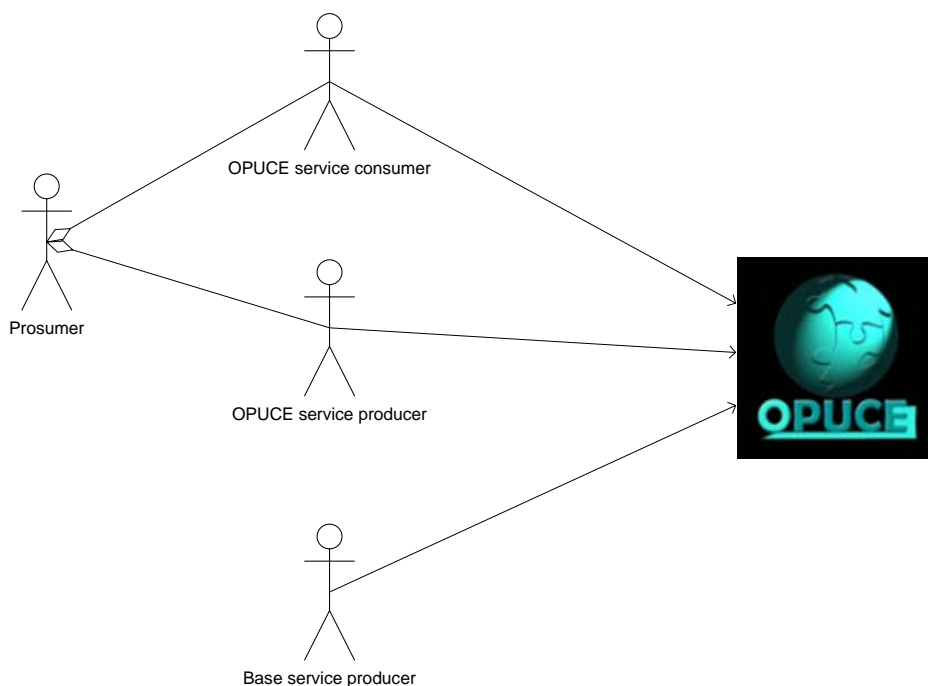


Figure 1. OPUCE platform users and roles

2.11. Verification and validation

Sometimes, the terms verification and validation are confusing, as they are used to mean the same thing even when their goals could be different. Within OPUCE, we will use the following definitions:

- **Verification.** The process of evaluating the OPUCE platform or its components or modules to determine whether they satisfy the requirements imposed at the beginning of the project (Requirements are specified in D2.1 i.e. this document). Verification could be done at a software unit level, module level or system level. Since requirements can be formally split into functional and non-functional requirements, the verification can follow this division too. In OPUCE, unit level and module level verification will be carried out by each unit development team. On the other hand, T5.3 will carry out system level verification.
- **Validation.** The process of evaluating whether the OPUCE platform is a working and useful product that conforms to user needs and intended uses, and that the particular requirements implemented through software can be consistently fulfilled. Since validation needs the end-user involvement, OPUCE will approach the set of target groups representing the different roles identified within this document. In OPUCE, T5.3 will conduct the end-user validation.

2.12. Service life cycle

Service life cycle is the process a service goes through from initial idea, via creation, deployment in the Service Provider platform, maintenance or evolution, until final withdrawal/undeployment from the Service Provider platform. In OPUCE the service life cycle will be considered as a combination of the management processes needed from the development stage to the withdrawal.

2.13. Service life cycle management

Service life cycle management is the sequence of steps and actions to perform state transitions from one Service Lifecycle state to another (including download, installation, configuration, provisioning, activation, and publishing), update (including download, installation, and activation), and removal/withdrawal of services performed through service lifecycle.

2.14. Service Creation Environment

The Service Creation Environment is a tool or set of tools that support the user to control the Service Lifecycle during its different phases, from the service design to the management of the service lifecycle, offering an integrated interface to the different subsystems.

2.15. Service Editor

The Service Editor is a tool that supports the service developer in the definition of the service logic. The Editor has a graphical interface to ease and guide the service definition process. The Editor is a part of the Service Creation Environment.

In OPUCE there are two different types of service editors, namely Web Editor and Mobile Editor. These editors were previously referred to as Advanced Editor and Basic Editor respectively, but in the second iteration of the project it was decided to give more intuitive and commercial names for the tools we are developing and using, so the names of both editors were updated.

The Web Editor is the tool we use for the definition of the service logic in PC's and is the more advanced version of the two service editors, hence its previous name. The Mobile Editor is a more restricted version of the Service editors, and it is aimed at being used in PDAs and other mobile devices that don't have as many capabilities as a PC.

2.16. Service Deployment

Service deployment is a sequence of processes/functions that get the service up and running and ready to be subscribed by end users. It begins after the service creation has been finished, and ends when the service is ready to be used by the subscribers.

The steps included in a deployment process are provisioning, instantiation, configuration, publication, and activation.

2.17. Service Undeployment

Service undeployment is a sequence of processes/functions that get the service down and withdrawn from the Service Provider platform. It begins with the service deployed in the platform (possibly running), and finishes when it is retired from the platform.

The steps included in an undeployment process are de-installation, denouncement, unprovisioning and deactivation.

2.18. Service Provisioning

Service provisioning refers to the "preparation beforehand" of IT systems' materials or supplies required to carry out a specific activity. It can be described using a composition script either using SPML or any other similar language. Service provisioning is a step of the deployment process.

2.19. Service Creation/Composition

Service composition is the ability, given to the service producer, to create, modify and save the specification of a service composition, defined as a flow of base services, using a set of base services that are available, according to the user's authorizations.

2.20. Service Simulation

Service Simulation is the chance, given to the service producer, to simulate locally the behaviour of a service flow they have designed. The simulation process takes place locally, without interacting with real network base services, and users have full control on the service flow (in terms of events raised, and actions performed) and service context.

2.21. Direct Service Advertising

It is considered a consumer driven action. A user of the platform finds a service that could be interesting for its friends, so it just sends the notification to them, i.e. everybody can send notifications to other OPUCE user about any service.

2.22. Service Sharing

It is considered a producer driven action. Service sharing can be performed under several scenarios:

- **First option:** The creator shares their service with people having some specific characteristics without knowing their names (user IDs), e.g. sharing a service with people living in Madrid, people that are above 18 years old, users with a specific IM client, etc.
- **Second option:** The creator of the service decides to publish the service and in this case a notification is sent to the users who included these service keywords in their preferences. This functionality was implemented for 1st iteration.
- **Third option:** The service has some keywords defined but there are no users who included these keywords in their preferences. When a user changes their profile including now some of those keywords or when a new user is created, the platform would also notify these users.

2.23. Service Recommendation

It is considered a platform driven action. The recommender recommends users with services that fit their life style. In order to do so, the recommender analyses the user habits (e.g. always uses communications services) and other parameters (e.g. number of users that use a specific service, if this number is big, then it seems to be a very popular service, so the platform also recommends it) and then the platform recommends those services to the user.

Note: the difference with the service sharing is that Service Sharing just uses the data of the service introduced by the creator (keywords) and also the preferences explicitly introduced by the user. On the other hand, as explained above, the Service Recommending, besides using keywords, is also able to infer by itself the characteristics of the service and the preferences of the users.

2.24. Personalization

Personalization is accomplished by adaptation, which in the context of OPUCE can happen at service creation, execution and any other step of the service lifecycle. In a user-centric platform like OPUCE, static information, like user profile, and more volatile data, usually named context, condition the user experience and allow to provide users with services tailored to their needs, preferences, interests or expertise.

Two ways of gathering user-related information such as profile or context can be distinguished:

- **Explicit.** It is based on direct feedback or input, no matter the user is aware of it (e.g. could be some application monitoring in the background, versus a web-based portal asking for user preferences (i.e. food, favourite movie types, etc.)
- **Implicit.** It is based on some post-processing (reasoning/learning) that deduce some information out of other explicit information

In general, personalization is expected to impact the modality to render the information as well as its prioritization, and eventually to automate part of the service invocation process, for example in pre-filling some input information or connecting various invocation steps.

2.25. Service Adaptation

In OPUCE, three different sorts of context aware service adaptation can be envisioned:

- **Context aware service composition:** the service creator, at creation/design time, decides how the service composition uses the service user context (i.e. at creation time is specified how the execution flow is driving using user context information).
- **Context aware base services (or component):** in this case, the context aware logic is inside the base services that compose the final service. These components are able to retrieve the user context, manage it and act in consequence having it into account.
- **Context aware implicit adaptation:** in this third type, the adaptation is performed automatically by the platform (implicitly), without the needed neither of creating the service using context aware base service nor of "drawing" the service flow using context data explicitly.

3. OPUCE user profile and context

3.1. OPUCE Users profiling

The aim of this section is to categorize the types of users that will be more eligible to use the OPUCE platform. User profiling consists of monitoring a user's specific behaviour in reference to a certain subject and then building a profile that describes the user's personal preferences on that subject. The obtained user profile will describe the user in terms of their age, gender and many other characteristics.

In OPUCE, the user profile that we obtain will be useful to get an idea of the acceptance our platform will have and it will also serve as input for Task 5.3 where these data is needed to define the profile of OPUCE users for their validation tasks.

As there are two different types of OPUCE users, the section is split in two. The first section targets OPUCE service producers and consumers, which are individuals that access OPUCE and create some services themselves or make use of existing services within the platform. The second section targets base service producers, which are organizations, or companies that create or develop base services on their own and then upload them to the platform in order to get revenues from it (e.g. every time their base service is used to create an OPUCE service).

3.1.1. OPUCE service producers and consumers

The following results have been extracted from the market study survey performed within MaCS project consortium [4]. Concerning Internet, the study tackles the following issues:

- Broadband penetration, as broadband is the base where multimedia services on the Internet stand
- Multimedia Services on the Internet, usage and user requirements and expectations

The broadband penetration as well as the use of multimedia services on the Internet gives us an idea of the parameters that will contribute in the use of the OPUCE platform by the users, including also here the creation of OPUCE services besides the use of already created ones. Let's see some of these data.

First data we get from the market study is the segmentation of individuals according to their communication profile: intensive users and less frequent users. In the segment of intensive users of telecommunication services (42% of population, 69% of flow), we can differentiate:

- **Mobile big clients:** among them, young people and teenagers are over represented.
- **Fixed phone intensive users:** strongly represented by women older than 50, non-active.
- **Big communicators:** using both fixed and mobile phones are represented by intermediary professions and 35-49 years old.

These data leads us to deduce that young people and adolescents, as mobile big clients, will be the ones to make a further use of OPUCE services, and they will also play an important role as creators, when the editor is enabled for mobile devices.

On the other hand, in the segment of less frequent users of telecommunication services (58% of population, 31% of flow), they are divided among:

- **Mobile little users,** among them working class and 25-30 years old are over represented.

- **Intensive users of fixed phone:** strongly represented by 25-34 years old with a low level of studies, and also modest working class and inactive people.
- **Little communicators of both mobile and fixed phones.** They also make an important use of SMS and mails. In this last segment, two individual profiles are over represented: children/teenagers and young parents.

In this case, we can also deduce that teenagers and young parents will be more likely the main users of OPUCE platform, since they are familiar with the use of SMS and mail and therefore, used to deal with computers and mobile phones.

Concerning services, the market study states that the average Internet user persistently spends time web browsing or sending and receiving e-mail messages. Both applications are the most prevalent among adults. P2P applications and Instant Messaging have quickly become popular among teenagers, who like to share lots of media files and to be in contact with friends at any time when they are at home.

So, from this data, we can extrapolate that in OPUCE, the main OPUCE service creators and consumers of the platform will be adults and also teenagers, for different reasons in each case.

As a good indicator of people getting on line to use the OPUCE platform, the number of people that use Internet to perform daily activities is increasing everyday. In Spain, for example, 41% of Internet users read on-line newspapers, 34% book tickets for the cinema or the theatre and 25% shop some products on-line.

The following results have been extracted from the 'Online Tool Use Survey' undertaken by the JISC funded SPIRE project [5].

The survey is on several Web 2.0 services, some of them are referred to using specific examples such as Flickr [6] and Wikipedia [7], whereas other services were referred to as a general type such as blogs and wikis.

The chart below shows the survey results in green (February 2007) and a prediction of what the results of a similar survey might look like in 2-3 years time. The flattening of the drop-off between 18 and 34 is the important aspect to consider. It is likely that the take-up among the young will remain strong and will ripple through as individuals move through the age groups.

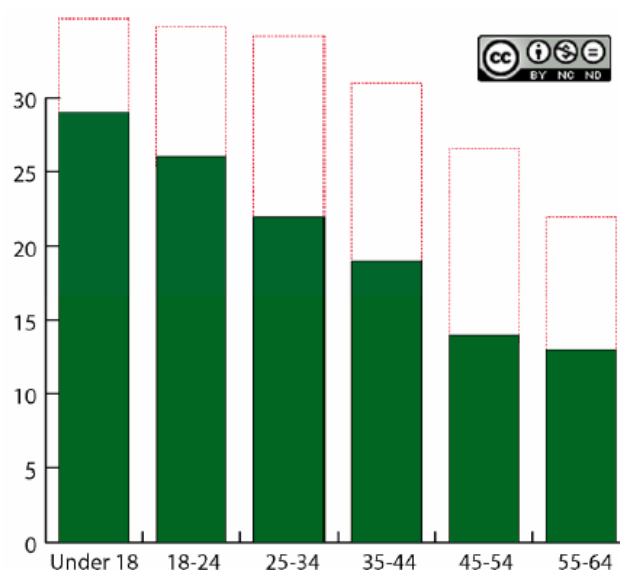


Figure 2. Overall percentage usage of tools/services across age

Respondents were asked if they contributed to each group of tools and services or simply viewed the material offered. But, we must bear in mind that the concept of contribution is subjective. Recent research in this area distinguishes between 'comments' and 'content creation'. Regardless of that, the survey results show an increase on the level of contribution compared to results in 2006. The survey indicates that 20% of people using mySpace [8] and youTube [9] contribute in some form.

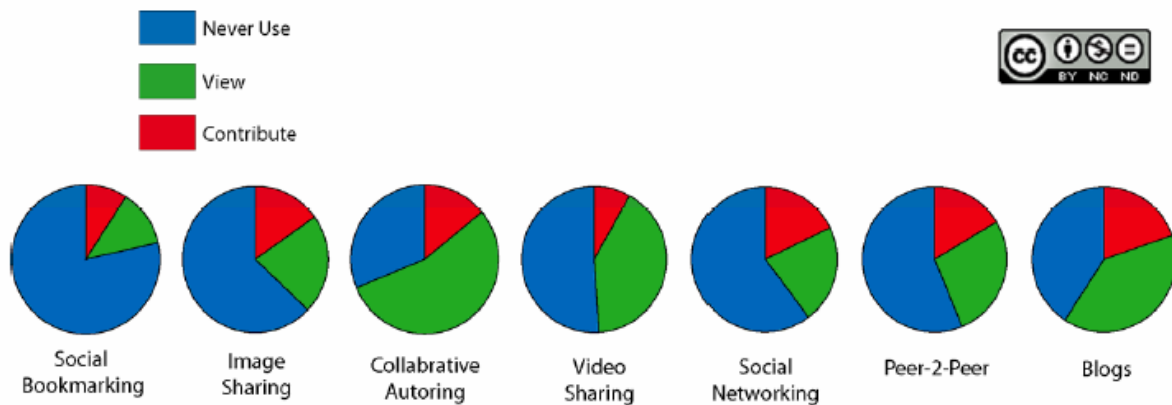


Figure 3. Ratios of contribution to viewing for groups of services

We get similar results form other studies, as in the case of Forrester [10]:

Once again we get results from studies on the use of broadband, in this case reflected as the percentage of content generated by others that the users watch or download from the Internet. These are the percentages for watching user generated content on line for each range of age:

- Under 18: 68%
- 18-24 age: 52%
- 25-34 age: 40%
- 35-44 age: 30%
- 45-54 age: 21%
- 55-64 age: 12%

From the same previous study, the percentages on groups of users of a certain age that upload video or audio created by them:

- Under 18: 21%
- 18-24 age: 13%
- 25-34 age: 10%
- 35-44 age: 5%
- 45-54 age: 2,5%
- 55-64 age: 1%

As we can see the user generated content is more popular among youngsters, and so we can estimate that similar percentages will be found for user generated services, as in the case of OPUCE.

To sum up, and based on previous data, this is the profile we should consider as users of the OPUCE platform:

OPUCE Service Consumers	30%	23 %	18 %	13 %	10 %	6 %
OPUCE Service Producers	40 %	25 %	19 %	9 %	5 %	2 %
Age	Under 18	18-24	25-34	35-44	45-54	55-64

Table 1. OPUCE Service Consumers and Service Producers per age

No important differences between genders have been encountered in the use of OPUCE platform. The studies show differences in the use of services among males and females but they strongly depend on the specific service so at this moment it is impossible to predict whether the platform will have more male or female service consumers and the same applies to service producers.

3.1.2. Base service producers

This section is aimed at identifying the profile of the organisations or enterprise groups, which we have defined as base service producers or third parties that will likely create base services for the OPUCE platform.

We have researched among some of the current initiatives in the IT world such as Google Android in the Open Handset Alliance [11] or the Telefonica Movil Forum initiative [12], to analyze what is the profile of the organisations that participate in these initiatives, since they can be identified as content providers.

These organisations, that are potential candidates to participate in the creation of services using the OPUCE platform, have a common factor: they offer innovative services and respond better to consumers' demands.

We have found the following characteristics that could help us to identify the profile of the OPUCE base service producers:

- leading providers of innovative software products
- providers of consumer oriented products, preferably companies oriented to wide range of customers
- organisations that want to strengthen their position as leaders and innovators for their customers
- global leaders in the deployment of technologies in mobile phones, computers and other consumer electronic devices
- important companies with an important amount of capital and revenues and hundreds of employees
- involved in business services: Research, development and sales of software for mobile and personal computers

- worldclass software providers, telecommunication, interactive multimedia, and consumer electronics service markets champions.
- business and consumer oriented solutions providers around the world.
- companies that want to offer productive and compelling products/services
- premiere developers of solutions found in consumer products worldwide, specially for young people or consumers of wireless devices, PCs, games, music software, etc.
- organisations that develop solutions to empower other companies success by providing them with certain solutions and expert services, with support around the world.
- companies that develop innovative, useful applications, in some of the following areas:
 - Social networking
 - Media consumption, management, editing, or sharing, e.g., photos
 - Productivity and collaboration such as email, IM, calendar, etc.
 - Gaming
 - News and information
 - Rethinking of traditional user interfaces
 - Use of mash-up functionality
 - Use of location-based services
 - Humanitarian benefits
 - Applications in service of global economic development
- organizations that belong to some of the following areas:
 - Educational centers
 - Developers
 - Consulting
 - Content providers
 - Device providers

As we can see from the results above, the profile of the so-called base service providers is the one matching with a leader organisation in its own specialized vertical segment and oriented to providing solutions to customers and businesses for their customers, and eager to maintain their position as leaders and innovators.

3.2. Terminals

This section aims at profiling the user terminals which have been envisioned in the OPUCE prototype. It includes some technical features which user devices must at least accomplish to be able to access to the Portal, the editors as well as to run services.

3.2.1. Service development

OPUCE platform offers to the user the opportunity to develop new service compositions from different terminal types: PC and mobile devices.

To develop new services from a PC, the minimum requirements are:

- Browser supporting Javascript: e.g. Firefox 1.5 or higher, Internet Explorer 6.0 or higher.
- A working web connection.

Mobile devices are heterogeneous devices that come in several screen sizes, resolution and capabilities. For service development on mobile terminal devices the OPUCE platform provides the Mobile editor. The editor has the following device pre-requisites:

- Devices: PDAs
- Processor speed: 400 MHz
- Screen resolution: QVGA (320x240)
- Screen sizes: All (due to adaptation features)
- Storage memory: 40 MB
- Program memory: 30 MB
- Operating System: Windows Mobile 5
- Active internet connection

Most of Windows mobile devices today fulfil these requirements, making OPUCE to support a broad range of these devices.

The choice for Windows mobile devices was made by the fact of the rapid increasing of devices using this OS. The use of .net applications will also allow the OPUCE applications in the future to run on Linux devices as there are some projects to implement the .net CF on Linux OS devices [13].

Finally there is also an article stating that “By 2010, Linux and Windows will both overtake Symbian in the market for advanced mobile phone OS, says a report from The Diffusion Group (TDG). Linux currently claims 23 percent of the market, while Symbian has 51 percent and Microsoft 17 percent, the research firm reports” [14].

Also Gartner said “Windows Mobile Devices Drove Worldwide PDA Market to 40 Percent Growth in First Quarter of 2007” [15]:

Company	1Q07 Shipments	1Q07 Market Share (%)	1Q06 Shipments	1Q06 Market Share (%)	1Q06-1Q07 Growth (%)
Windows CE	3,184,703	62.1	1,937,667	52.8	64.4
Research In Motion	928,239	18.1	929,883	25.3	-0.2
Palm OS	314,353	6.1	489,220	13.3	-35.7
Symbian	288,000	5.6	132,000	3.6	118.2
Linux	33,400	0.7	43,530	1.2	-23.3
Others	377,150	7.4	137,000	3.7	175.3
Total	5,125,845	100.0	3,669,300	100.0	39.7

Table 2. Worldwide: Preliminary PDA Vendor Shipment Estimates by Operating System.

Note: Excludes Smartphone's, such as Treo 750 and BlackBerry 81xx, but includes cellular PDAs such as BlackBerry 87xx. [16]

Service development will be done on data driven mobile devices (PDA) relegating other devices as Smartphone's or mobile phones to service usage. It is stated that OPUCE is supporting a large range of devices.

3.2.2. Service usage

Device requirements for service usage are strongly influenced by the type of base services that are involved in a service composition. The target devices of OPUCE service can range from ordinary mobile devices to PCs.

If internet based services are consumed, the same device requirements as service development will apply.

3.3. User context

Human-computer interaction is an important concept that does necessary the introduction of context aware computing which idea is to adapt the computer and its behaviour to the user context.

There are a lot of definitions of user context but, in general, we can use user context to “any information that can be used to characterize the situation of an entity (person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves)” [17]

The user context is especially dynamic for mobile systems such as mobile phones, laptop, etc. When using such devices, it is important to adapt the application's behaviour to the ever changing situation.

Some important aspects of context are [18]:

- **Environmental Factors.**

There are different approaches to choose the relevant aspects in environmental context. For example, in [18], it is distinguished between comprising network variations (bandwidth, latency, etc.), hardware variation (screen size, buttons, etc.), and memory and software variations (memory capacity, installed applications, etc.). In [18], the information about the infrastructure and the location information (physical and logical, at home versus at work, e.g.) is also depicted as very relevant for the user context.

The following kinds of environmental context can be considered:

- **User Context:** This aspect takes into account personal information about the user and user's classes as user's identity, characteristics, capabilities, universal preferences, the state of the user, information about his or her main activity, etc.
- **Resource Context:** The resource context refers to the demand of multi-delivery and thus includes information about the relevant devices, device classes, documents, network, available services, etc.
- **Location Context:** This context describes the geographical coordinates, identity and the state of the location (the people present at that location, etc). The location context

includes also aspects of the perception of the physical characteristics of the location, e.g. temperature, brightness, noise levels, etc.

- **Temporal Context:** The temporal context (in diverse forms, e.g. the absolute time, hour, am / pm, etc.) allows to adapt the application with regard to certain timing constraints such as the hour to go to work.

General context categories	Features
Resources	Size of a display Type of the display (colour, etc.) Input method (touch panels, buttons) Network connectivity Communication cost and bandwidth Nearby resources (printers, displays)
User	Tasks of the user User's profile (experience, etc.) People nearby Characters, date and time formats
Location	Position (latitude and longitude) Lighting, temperature, weather conditions, noise levels Surrounding landscape User's direction and movement
Temporal	Time of day Week, month Season of the year

Table 3. Categorization of environmental context

- **Application State Factor.**

This factor has the information about the state of the application itself and complements the environment factors.

- **History Factor.**

This is necessary to identify if the context values change over time. For this, it is needed to consider the historical circumstances and the current ones.

- **Static and Dynamic Context.**

User information can be static or dynamic. Some context information is considered static if the context is defined and it does not change during the application execution. The context information is considered as dynamic if it is determined during run-time.

- **Availability of Context Information.**

Not all context factors are always available, e.g. if one user has not interacted with the system yet, there may be no data about him/her available.

3.3.1. User Context in OPUCE

The user context that we consider in OPUCE is a very wide, open concept, as there are no restrictions in the user context to be considered. Following previous guidelines, we are going to describe the OPUCE user context through its different factors:

- **Environmental factors:** In OPUCE, environmental factors regarding user context are very important since the platform will contain a wide range of services to be executed in different environments. As explained before there are several kinds of environment contexts, which are explained below in relation to the OPUCE platform.
 - **User context:** In OPUCE, the user context information is composed of two parts: one static part (such as address, name, preferences), that is stored within the User profile and another dynamic part (such as location), which is collected at each moment by specific clients in the end-device and sent to the platform to be stored there. The user context will help the platform to adapt to the user preferences and it will also be used for identification tasks.
 - **Resource context:** In OPUCE this context is represented by the list of available services and the type of terminal being used. This last parameter will indicate the platform whether to use the Web Editor or the Mobile one.
 - **Location context:** In OPUCE this information is represented by the location provided by the Platform after having processed the row data collected from the user device. Some services will take into account the exact position of the user (latitude-longitude), which will change as the user moves, so in this case the information will not be obtained from the user profile but from a base service providing this functionality.
 - **Temporal context:** In OPUCE some services will be executed at a determined time, being the temporal context. The user specifies the temporal constraints of a determined service if that service offers that possibility of personalization.
- **Application state factor:** The state of the OPUCE application is represented by a workflow engine.
- **History factor:** The history factor is not taken into account in OPUCE.

In OPUCE, some of the context info will be static, such as the information contained in the user profile, and some other will be dynamic, such as the user location and resources, which is stored in the proper module of the platform.

4. Use Case description

This section introduces several use cases that represent the main possible actions to be performed using OPUCE platform. These actions have been divided into three high level use cases and then there will be lower level use cases within all of them. High level use cases represent each user role that is going to be involved with the platform, i.e. service producer, service consumer and base service producer.

4.1. An OPUCE story

The following table presents a simple story which explains how the OPUCE platform is used to create, manage and execute services. It tries to link in a single scenario the low level use cases that will be explained below. The role of the user who is involved in the use case is also specified.

Story	Low level use case	User role
Alice wants to create a new service including IT and telco facilities by using OPUCE platform. Alice accesses the Web editor through the OPUCE portal or the Mobile editor and composes the service (S1) by linking base services and configuring parameters. - Cont (*) -	Service composition	Service producer
Alice selects simulation view in the editor and checks that S1 runs properly.	Service simulation	Service producer
(*)S1 is stored in the Service Repository associated to keywords K1, K2 and K3.	Service composition	Service producer
S1 wants to share S1 with their workmates (Carol and Dave) and selects those contacts in the social network facility to get notified. Carol and Dave are announced about S1 release.	Service sharing	Service producer
Bob wants to search in the OPUCE platform in case there is a service already created that can be useful for them. Bob accesses the portal and the list of existing OPUCE services is shown.	Service look-up	Service consumer
Bob chooses the service S2 that better fits the user's needs and subscribes to it. Bob goes to OPUCE portal or accesses the application deployment widget in their mobile device; configures the open parameters and runs S2 on the platform. During usage, the service is adapted to the context of Bob.	Service subscription, personalization and usage	Service consumer
Bob loves S2 and wants to advertise about it to some of their friends (Eve and Frank). For this, Bob uses the social network	Direct service advertising	Service consumer

functionality that OPUCE platform offers. Eve and Frank receive notifications about S2 through the preferred way (SMS, e-mail, etc)		
Bob changes their preferences about services which could be interesting for them, including keyword K2. Immediately the platform launch the process to match users and services, and Bob is notified about S1.	Service Recommending	Service consumer
Bob subscribes to S1 and when deploying Bob receives on their terminal a notification requesting the installation of some software which is required to execute S1.	Remote Service Management	Service consumer

4.2. OPUCE platform elements

In order to improve the results obtained during the 1st iteration, the description of the use cases have been done considering specific modules of the platform. Such modules were identified in T2.3, WP3 and WP4 during the 1st iteration and will be refined during the 2nd.

The top level platform modules and submodules which have been used in the description of the use cases are listed below. The functionalities which are accomplished by the top level modules are explained in section 2, therefore, descriptions are not included to avoid duplicating information

- **Portal**

- *Web Editor*. It is a web application that provides the interface to be used by the service creator, to compose services combining functionalities via a graphical interface.
- *Mobile Editor*. It is an independent application that runs on a mobile device that allows the users to create service compositions, with the constraints of those platforms.
- *Component Editor*. It allows trusted 3rd parties to create and save OPUCE base services, starting from a set of software modules previously created, editing component description (both master and facets description) with a friendly user interface and to manage them (deploy, undeploy, monitor).

- **Service Lifecycle Manager**

- *Service Repository*. It is a database maintaining the descriptions of service compositions. It offers interfaces to create, retrieve, update and delete specific facets or the complete descriptions. Complex search operations are offered in the same way as simple notification mechanisms.
- *Component Repository*. It is a database which contains the descriptions of base services or components. In the OPUCE prototype, the same database has been used to store service compositions and base services, and it is usually referred as Service Repository.

- *Deployment Manager*. It orchestrates the transition between the service life-cycle phases and co-ordinates the adjacent activities mandatory in the SLM and the SEE in order to provide the operational service in the execution environment. Since part of the modules may be located on the client device, specific mechanisms for *remote Service Management* has to be offered (**Deployment Manager submodule RSM**).
- **User Information Manager**
 - *User Profile Repository (UPR)*. It is a storage that takes charge of user-related data (e.g. user profile, user preference, user service subscription).
 - *History and Learning Engine (HLE)*. It is the element inside the platform that collects all the service usage data, and tries to use those data to extract situation, behaviours and preferences, to add dynamic information to the users' profile.
 - *Context Enabler & Reasoning (CER)*, is in charge of gathering and providing context information or information related to matching contexts
 - *Context Usage and Feed (CUF)*, sub-block that collects service usage and context data from end-users and makes them available over the User Information Management subsystem (UIM).
- **Service Advertiser**
 - *Matcher*
 - *The Matcher Broker (MB)*, triggered by any changes inside the Service preferences of the user profile, use the right Matcher Provider (MP) to identify new services to recommend the user.
 - *Matcher Providers (MP)*, are the real logics inside the Matcher (Framework) able to match user-services, using their own algorithms. In the OPUCE Platform there are the Semantic MP and the Collaborative Filtering MP.

4.3. OPUCE service consumer Use Cases

This section presents the OPUCE user performing the role of a service consumer. So it is assumed that the user, besides being only a consumer or a Prosumer (producer and consumer), will only execute consumer actions. Following sections describe a series of lower level use cases that are part of the high level service consumer use case.

4.3.1. Service look-up

4.3.1.1. Use case description information

- **Name:** Service look-up.
- **Goal:** The user searches inside the portal for the desired service. The portal shows the list of existing OPUCE services, sorted by date and alphabetically. The user can look for a service by searching through the list or by introducing a related word in the

corresponding field so the platform can execute the search and respond the user with the service or services related to the word they previously introduced.

- **Use case team leader/members:** TID / UVA, TIL, PTI.
- **Pre-condition:** The list of existing OPUCE services must contain some services so the user can execute the search. The user accessing this functionality of the portal must have been granted with the privileges to do so (just registering in the platform would be enough).
- **Post-condition:** The platform provides the user with the list of services that are related to the word the user introduced in the search field.
- **Constraints/Issues/Risks:** If the Portal is down or the search engine is down, then the user won't be able to execute the search.
- **Trigger event(s):** The user accesses the portal and the list of OPUCE services.
- **Primary actor:** OPUCE service consumer.
- **Secondary actor(s):** None.
- **Components involved:** Portal, service repository.

4.3.1.2. Use case detail

- **Pathway name:** OPUCE service look-up.
- **Trigger event:** The user accesses the list of OPUCE services within the portal.
- **Main sequence of steps:**
 1. The service consumer accesses the OPUCE portal.
 2. Access to the Portal is granted in case service consumer credentials are correct and sufficient.
 3. The user introduces one or more words related to the kind of service the user wants to find.
 4. The platform executes the search and shows the list of found services to the user, filtering those that the user is not authorised to use.
 5. The user then accesses these services related information, such as service description, so the user gets the exact idea of what each service does and decides whether to subscribe to these services or not.
- **Variations:**
 - In case such service needed by the user does not exist, then the platform will show no services after executing the search.
- **Extensions:**
 - *Incorrect or insufficient credentials:* either if authentication fails, in step 2 access is not granted and use case is aborted.
- **Business rules:** None.
- **Special requirements:** None.
- **Constraints/Issues/Risks:** If the Portal is down or the search engine is down, then the user won't be able to execute the search.

4.3.2. Service subscription, personalization and usage

4.3.2.1. *Use case description information*

- **Name:** Service subscription, personalization and usage.
- **Goal:** This process includes all the steps needed in order to execute a service available in the platform. The process may include the following steps: subscription to the service, provisioning of some user information and preferences to personalize such service and finally service execution, including adaptation to user context.
- **Use case team leader/members:** TID / UVA, TIL, PTI
- **Pre-condition:** OPUCE user has a profile stored in the platform, created when the user joined the community. The OPUCE service must be already created and deployed in the platform, with a list of open properties set at creation time to allow personalization. The user is logged into the platform.
- **Post-condition:** The service is properly subscribed and the execution started. All the open parameters have been filled so the service is personalized to user's tastes.
- **Constraints/Issues/Risks:** None.
- **Trigger event(s):** The user wants to subscribe to an OPUCE service.
- **Primary actor:** OPUCE service consumer.
- **Secondary actor(s):** None.
- **Components involved:** Portal, Service Repository, User Profile Repository.

4.3.2.2. *Use case detail*

- **Pathway name:** Service subscription, personalization and usage.
- **Trigger event:** The user wants to subscribe to an OPUCE service.
- **Main sequence of steps:**
 1. The user discovers an interesting service by whatever means it has considered appropriate (notification, browsing, service gallery, etc).
 2. The user decides that they want to use it so the user starts the process of subscription through the corresponding interface in the Portal.
 3. The platform checks that the user is authorized to subscribe and use the service
 4. The portal has to check which configuration parameters are needed for the execution of the service. This information is stored in the service description, which is retrieved from the Service Repository.
 5. The Portal automatically creates a configuration page to be presented to the user with the subscription parameters.
 6. The user sets the subscription parameters of the service, which are parsed by the Portal and sent to the User Profile Repository to be stored inside the OPUCE user profile.
 7. The user queries the execution of the service.

8. The Portal again with the Service Repository to retrieve the service parameters list. The Portal presents the form partially filled with the subscription-time parameters and the empty parameters still to be set by the user.
 9. The user sets the empty parameters and/or changes those configured at subscription time and activates the service.
- **Variations:**
 - The user is not subscribing a new service, but changing the personalization information of a previously subscribed one. In that case, between steps 4 and 5, the Portal retrieves the previous configuration data from the User Profile Repository.
 - The subscription parameters are already configured. The Portal does not create a configuration page and passes from step 4 to step 7.
 - There are not empty parameters when the user queries the execution. A "Run" button appears directly.
 - **Extensions:** None.
 - **Business rules:** None.
 - **Special requirements:** None.
 - **Constraints/Issues/Risks:** All elements listed in the use case sequence must work together.

4.3.3. Direct Service Advertising

4.3.3.1. *Use case description information*

- **Name:** Direct Service Advertising.
- **Goal:** Service end-user recommends the service to one or more elements from the user's social network.
- **Use case team leader/members:** TID / UVA, TIL, PTI.
- **Pre-condition:** It is not necessary that the end-user is subscribed to the service. The end-user is logged on into OPUCE User Portal and is using some service portlet (widget) exhibiting a "Service Sharing" tab. The user's OPUCE Social Network is not empty.
- **Post-condition:** Advertised end-users receive notifications of the advertised service.
- **Constraints/Issues/Risks:** Notifications should not be sent to end-users that are already subscribers to the advertised service. To avoid spam, notifications should only be allowed to users from Advertiser's Social Network with an authorisation rule allowing recommendations from the Advertiser End-user.
- **Trigger event(s):** The end-user pushes the "Service Sharing" tab.
- **Primary actor:** OPUCE service consumer.
- **Secondary actor(s):** End-users receiving notifications to subscribe the advertised service.
- **Components involved:** Portal, User Profile Repository, Service Advertiser

4.3.3.2. Use case detail

Direct Service Advertising use case is basically the same as Service Sharing use case (section 4.2.4). The only difference is the primary actor, which in the Service Sharing is the owner of the service, instead a service consumer as in the Direct Service Advertising use case. For this reason we avoid giving the use case details here, as they are also described in the Direct Service Advertising use case, with the only commented difference that the main actor is now the service consumer.

4.3.4. Service recommendation

4.3.4.1. Use case description information

- **Name:** Service Recommendation
- **Goal:** Recommend and notify one or more services to the OPUCE users using their profile, their context and all the usage history.
- **Use case team leader/members:** TID / UVA, TIL, PTI
- **Pre-condition:**
 - Every OPUCE user has **a profile** stored in the platform (User Profile Repository), created when the user joined the community;
 - One or more **sensors** inside the end-user/platform collect and store all the usage information (e.g. when a service is run, who runs it, which is the user context, etc) in the platform.
- **Post-condition:** A notification that contains the list of the recommended base services or composed services is sent to the user.
- **Constraints/Issues/Risks:** None
- **Trigger event(s):**
 1. The service preferences section of one user profile changes. More than one reason is possible :
 - A new user has joined OPUCE (a brand new profile is created)
 - The user modifies their profile (preferences)
 2. Changes in the of a “service profile” done by the service creator
 3. The Recommendation process can be directly started by the user.
- **Primary actor:** The platform that triggers changes in the profiles
- **Secondary actor(s):** The OPUCE user receiving the recommendation
- **Components involved:** User Profile Repository, Matcher (Matcher Broker and Matcher Providers), Service Advertiser, Portal, History and Learning Engine, Context Storage, Usage sensors inside the platform (SEE, Portal, end-users)

4.3.4.2. Use case detail

- **Pathway name:** Service Recommendation (profile based)

- **Trigger event:**
 - The preferences section of a profile changes (new or updated) inside the UPR. This could happen for different reasons:
 - A new user joined OPUCE: from the Portal the user fills the registration form, and compiled the Profile section. At the end of the registration process a new profile is stored in the UPR.
 - The user from the Portal or from a client modified their preferences.
 - The learning machine (in the HLE) dynamically updates the user preferences (after a user confirmation), using the service usage history.
 - The service profile is modified by the service creator.

- **Main sequence of steps:**
 1. The User Profile Repository notifies the Matcher Broker about any updated profiles. (Only the preferences section of the Profile is taken into account).
 2. Using the available information, the Matcher Broker selects the right Matcher Provider to start the recommendation analysis.
 3. The Matcher Provider uses its own algorithm (Collaborative Filtering, Semantic, indexing, etc.) to select all the services which descriptions fit with the user preferences.
 4. The resulting services list is sent as input to the Service Advertiser.
 5. The Service Advertiser notifies the user in the preferred way.

- **Variations:** The user can directly require a Service Recommendation, by using the Portal where a specific section (Portlet) can trigger the recommendation process. Also clients or Context Enabler can trigger special situations or events where they can start the process.

- **Extensions:**
 - The Matcher Broker may use more than one provider and then it aggregates the results.
 - *No Service founded:* The Matcher providers can't find any service to recommend. The recommendation process is aborted here.
 - *No suitable services found:* All the services identified by the Matcher providers are already subscribed by the user. The recommendation process is aborted here.

- **Business rules:** None

- **Special requirements:** User Profile Repository must implement a notification mechanism to notify the other component when a profile is updated (e.g. it notifies to the Matcher Broker)

- **Constraints/Issues/Risks.** None

4.3.5. Remote Service Management Use Case

4.3.5.1. Use case description information

- **Name:** Remote Service management
- **Goal:** After a service subscription, done by the user, the Remote Service Management deploys all the needed elements on the Service Execution Environment and on the User End Device (ED).
- **Use case team leader/members:** TID / UVA, PTI, TIL.
- **Pre-condition:** The end-user is logged into the OPUCE User Portal and has the RSM Management Widget installed on its end device.
- **Post-condition:** All the needed elements are deployed automatically in the platform and on the End Device of the user (Widgets).
- **Constraints/Issues/Risks:** The self provisioning task is done in two steps. First, the RSM collects and installs all the elements in the platform. Then (step 2) it sends a SIP notify to the end device to perform the remote provisioning. The user will receive it and will decide to install the element/s immediately or to postpone this task. This last case could generate problems if the user tries to run the service before the user hasn't already completed the installation process (on the ED).
- **Trigger event(s):** The end-user subscribes to a service
- **Primary actor:** Service consumer
- **Secondary actor(s):** None
- **Components involved:** Portal, User Profile Repository, Deployment Manager, Deployment Manager submodule RSM, Remote Service Management (RSM) widget, IMS SIP Proxy (external).

4.3.5.2. Use case detail

- **Pathway name:** Remote Service Management.
- **Trigger event:** The end-user subscribes to a service on the Portal from the user's service catalogue page.
- **Main sequence of steps:**
 1. The Deployment Manager receives the subscription event from the portal and finds that one or more of the base services used are remote services. Therefore it retrieves the necessary facets and triggers the RSM platform module.
 2. If any component is needed on the End Device, the Deployment Manager submodule RSM gets its SIP contact from the user profile, in the User Profile repository.
 3. Deployment Manager submodule RSM sends a SIP message to the user through the IMS SIP Proxy. The message contains the description of each element that needs to be installed (and how to get them).
 4. The notification is received by the RSM widget on the End device.
 5. The RSM widget displays a pop up message and adds an item (e.g. INSTALL component A) on the graphical interface of it.

6. At this step the user can do two things:
 - Install immediately the new components: The widget automatically downloads the components and installs them in the device. If all the process is successfully terminated the widget sends back an OK message. Otherwise, a failure message is sent back to the Deployment Manager submodule RSM.
 - The user postpones this task: the notification remains as a remark in the Graphical interface, and the Deployment Manager submodule RSM receives back a postpone message. But it knows that the delivery message was correctly received.
 7. The Deployment Manager submodule RSM waits until all the elements are correctly installed (Platform/Client side) to enable the service to be run!
 8. The Service is now able to be run.
- **Variations:** None
 - **Extensions:** None
 - **Business rules:** None
 - **Special requirements:** The User Profile has to contain the SIP contact of the user.

The end user must have some things in the End user device:

- The Yahoo! Widget Engine.
- The RSM widget.
- The user must be (SIP) registered on the platform in order to receive the RSM notification.

4.3.6. Context Usage and Feed Use Case

4.3.6.1. *Use case description information*

- **Name:** Context Usage and Feed
- **Goal:** Gather raw data from the end user device to discover the user's context and service usage. This information is (temporally) stored and processed by the UIM that then makes it available for all the other components of the platform.
- **Use case team leader/members:** TID / UVA, PTI, TIL.
- **Pre-condition:** The user has to install client(s) in their end-devices to collect raw data.
- **Post-condition:** All Context and Usage information is stored in the UIM. After a backend process the UIM makes them available to any platform elements.
- **Constraints/Issues/Risks:** Different clients are available to perform this task. Depending on the implementation and the OS where they run, different subset of data should be collected. For example in the mobile client it is possible to get raw data that will be used to get the location. On the Client side this scope must be simulated (by now).
- **Trigger event(s):** The user installs and runs a Context Usage and Feed client on their device.
- **Primary actor:** Service Consumer
- **Secondary actor(s):** None

- **Components involved:** CUF, CER.

4.3.6.2. **Use case detail**

- **Pathway name:** Context Usage and Feed.
- **Trigger event:** User installs and runs a Context Usage and Feed Client on their device.
- **Main sequence of steps:**

1. A CUF client is installed on user's device

The first time the CUF is started a configuration process is needed. The CUF needs at least the USERNAME of the USER, to post the information on the OPUCE Platform.

- The User can decide if some information must not be sent.
- When CUF is running it tries to collect all the available raw data from the device. This depends on the logic on which the CUF is done. Some examples of raw data could be the following: Memory, OS, bandwidth, location, BT nearby.
- About Usage, the CUF tries to get all the information of the other widgets downloaded and used in the Yahoo engine, and the Browser History if available.

CUF envelops the data in a XML document called "ContextML" . The different data collected are split in ContexteElement: Each of them groups row data by scopes and provides a description that includes when it was got, how long it is valid and other information.

2. CUF sends the whole document to the CER via HTTP post.
3. CER receives it and stores the available data in the DB.

Each scope has an expiration time (defined in the *ContextML* Document) and it defines the availability of that scope in the CER.

- **Variations:** None
- **Extensions:** None
- **Business rules:** None
- **Special requirements:** None

4.4. **OPUCE service producer Use Cases**

This section of the Use Cases collection presents the OPUCE user performing the role of a service creator, besides the same user acting as a consumer of theirs or other's services. The objective of this section is to list only the creation related activities.

4.4.1. Service composition

4.4.1.1. *Use case description information*

- **Name:** Service composition.
- **Goal:** Create and save a service composition, starting from a set of available base services.
- **Use case team leader/members:** ERI / TIL, HUA.
- **Pre-condition:** The user is already registered in the OPUCE Portal. The user is logged into the OPUCE platform. There is a set of available base services.
- **Post-condition:** A new service composition is created and stored in the repository.
- **Constraints/Issues/Risks:** There is the need of a Web Editor, which allows a graphical and user-friendly composition with only a web browser, and of a Mobile Editor, which is a .application (previously downloaded), to allow composition from PDAs.
- **Trigger event(s):** The OPUCE service producer accesses OPUCE Portal (for Web Editors) or the OPUCE service producer accesses directly the Mobile Editor installed application.
- **Primary actor:** OPUCE service producer.
- **Secondary actor(s):** None.
- **Components involved:** Portal, Web Editor, Mobile Editor, Service Repository.

4.4.1.2. *Use case detail*

- **Pathway name:** Service composition.
- **Trigger event:** The OPUCE service producer accesses the OPUCE Portal (for Web Editor) or the Mobile Editor directly.
- **Main sequence of steps for the Web Editor:**
 1. OPUCE service producer creates a new service, inserting their general information (e.g. the service name).
 2. OPUCE service producer decides to edit the newly created composition and they are redirected to the Web Editor.
 3. The palette of the Web Editor is filled with the base services (split in categories) authorized for the registered user.
 4. OPUCE service producer selects the starting base service for the new composition from the palette by clicking on it: the base service is shown on the canvas.
 5. Then the user decides to insert another base service in the new composition, so, they select another base service by clicking on it. As a result, the base service is shown on the canvas.
 6. Now the OPUCE service producer must insert the connection between the two base services deciding which event of the first is connected to which action of the second.
 7. OPUCE service producer must set the properties for each block (options for setting property values are: typing values directly, picking values from a list, inserting

references to values exposed by other base services in the service composition, inserting references to values from the user profile (\$me)).

8. OPUCE service producer can repeat steps 5, 6 and 7 inserting other base services, drawing connections and setting properties.
9. The user may decide to remove an already inserted base service from the composition and also to delete the connection between two base services.
10. At last OPUCE service producer must choose at least one final event for their composition.
11. When the user is satisfied with the composition and decides to save it, so the Web Editor saves it in the repository.

- **Main sequence of steps for the Mobile Editor:**

1. OPUCE service producer creates a new service composition.
2. OPUCE service producer selects the starting base service for the new composition from the list of base services.
3. Then the user decides to insert other base services in the new composition and they select another base service from the base service palette.
4. The Mobile Editor performs by default automated linking between actions and events. This procedure is triggered when a base service is added; its actions are matched against the events of the other base services to find the best possible option to a link trying to predict the user intentions. This procedure uses the events and actions specific properties' set and used respectively allowing the editor to decide which events are most probable to be connected to a specific action. All automatic links must be later validated by the user in order to be definitive and saved.
5. The OPUCE service producer may change the default automated linking, deciding which event of one base service is connected to which action of another one.
6. OPUCE service producer must set the properties of each block.
7. OPUCE service producer can repeat steps 3, 4, 5 and 6.
8. The user may decide to remove an already inserted base service from the composition and also to delete the connection between two base services.
9. At last OPUCE service producer must choose at least one final event for their composition.
10. When the user is satisfied with the composition, they decide to save it: so the Mobile Editor saves it in the repository (if it has connection with the repository) or stores it in the local cache.

- **Variations:**

- *Existing OPUCE service edition:* OPUCE service producers may choose to edit a previously created service. The existence of the selected service becomes a precondition. This is possible both from the Web Editor and from the Mobile Editor.
- *Cross editing:* A composition created with Web Editor can be edited by the Mobile Editor and vice versa, a composition created by the Mobile Editor can be edited also in the Web Editor.

- *Internationalization*: A composition can be edited in any supported locale, regardless of the one used at service creation time; all language-sensitive strings are rendered according to the preferred language of the OPUCE service producer.
- **Extensions:**
 - *User leaving the editor*: If OPUCE service producer decides to leave without saving the composition, they must be given the option to discard all the changes made or to save them.
 - *Missing final event in the Editors*: The Web and Mobile Editors will warn the user if the final event is missing.
- **Business rules**: All registered users will be able to compose services.
- **Special requirements**: Mobile Editor runs on PDAs with limited graphical capabilities, resources (memory, processor, storage, etc.) and connectivity meanwhile Web Editor runs on PCs or devices with advanced graphical capabilities.
- **Constraints/Issues/Risks**: A web browser supporting *JavaScript* and *Ajax* is needed for the Web Editor; and a PDA with the Mobile Editor installed for the Mobile Editor. Mobile editor will not support a very large palette of base services due to storage constraints, so in a very large base service palette the user would only have access to a sub-set of it when composing in the Mobile Service Editor in relation to a full palette at the Web Service Editor.

4.4.2. Service management

4.4.2.1. Use case description information

- **Name**: Service Management
- **Goal**: Service management is part of the service lifecycle process and excludes the service creation. Service management includes the process that allows end users to deploy (provisioning, registration, publication, testing and activation), withdrawal (undoing the steps done while deploying the service except the testing) and monitoring(info collected by the SEE/Metrics) of services already created by the end user.
- **Use case team leader/members**: ERI / TIL, HUA.
- **Pre-condition**: There will be a service that has been previously created, and the user will be able to manage it by doing the deploying, the monitoring and in case the service is already deployed, remove it.
- **Post-condition**: Service lifecycle as it is being considered in OPUCE has several steps. Within this use only deployment and undeployment have been included; with, different post-conditions each. If the service is deployed at the end of the process the system will have available a new service to be used. If the service is withdrawn (undeployed), the service will be removed from the system and will not be available to be used any more
- **Constraints/Issues/Risks**: None
- **Trigger event(s)**: After a successful service creation process, the service composer can trigger the service deployment through the OPUCE portal. The service owner is also responsible for the service removal, this removal can be activated both scheduling the task or directly by the service creator. Regarding service monitoring, the SEE will collect statistics of the service and a triggering event from the user will allow them to use this information.

- **Primary actor:** Service producer.
- **Secondary actor(s):** None.
- **Components involved:** Event Manager (events that fire the transitions in the state machine), Activity manager (this entity will manage all the activities that a state needs to achieve), Portal, Service repository, Service registry, Deployment manager, Provisioning manager.

4.4.2.2. *Use case detail*

- **Pathway name:** Service deployment
- **Trigger event:** Once the service has been created the service creator proceeds to deploy (provision, registration, publication, testing and activation) it. The service creator will trigger this process and the final output will be the deployed service.
- **Main sequence of steps:**
 1. Service composer composes a service and proceeds to deploy it (the triggering event).
 2. *Service Provisioning.* Service provisioning is carried out automatically contrary to traditional provisioning which often requires a manual participation to create accounts for users, reserve resources, etc. There are three provisioning tasks identified, each one affecting different elements of the OPUCE architecture:
 - *Component Provisioning*, which considers the provisioning tasks to be done in those base services that the user creator has combined. These activities include the reservation of resources or configuration of permissions to use the base services.
 - *Platform Provisioning*, which considers the provisioning tasks to be carried out in the OPUCE platform services such as updating billing systems and updating service repository. These tasks are common for all the created services.
 - *User Provisioning*, which considers the provisioning tasks to be carried out on the user side at subscription time, such as installing a dedicated application in the user's mobile device.
 3. *Service Registration.* In this activity the platform registers all the information needed to access the service once activated, such as endpoints if it is a Web Services-based access.
 4. *Service Publication.* In this activity the service is officially published including all associated attributes, e.g. service type, descriptions, terms and conditions of use, etc, making service discovery easier. Some of these data are taken from the service description, such as the service name, the semantic facet, etc. Publication is done via a service advertiser to which other users can subscribe specific keywords describing the service instance like "intelligent email", "email forward" or "email reading".
 5. *Service Test.* The aim of this activity is to ensure that a new service is ready to be subscribed to and consumed by end-users. Nonetheless, this activity has not been considered in the first stages of the OPUCE project. It might be included in the second iteration of the project.

6. *Service Activation*. This is the last step of the service deployment process. Once the service has been activated it becomes publicly available and ready for subscription. This activity is triggered by using the information included within the service lifecycle schedule facet of the service description.
 7. Service is available
- **Pathway name:** Service deployment
 - **Trigger event:** The service creator triggers the service undeployment.
 - **Main sequence of steps:**
 - The service removal consists of undoing the steps carried out while deploying the service (except for testing the service).
 - **Variations:** There are cases in which there is a deployment phase also executed on subscription. This is needed in case there are base services that are running on the end user device. These base services can only be deployed, when the user subscribes for the service. This constitutes a new type of use case, the Remote Service Management use case, which is further explained in section 4.3.5.
 - **Extensions**
 - **Business rules:** All registered users will be able to deploy services.
 - **Special requirements:** None.
 - **Constraints/Issues/Risks:** None.

4.4.3. Service Simulation

4.4.3.1. Use case description information

- **Name:** Service simulation.
- **Goal:** Enable the service producer to simulate locally the behaviour of a service flow the service producer has designed.
- **Use case team leader/members:** ERI / TIL, HUA.
- **Pre-condition:** The user is editing a service composition.
- **Post-condition:** The user has been able to check the simulated flow, and eventually can resume service edition.
- **Constraints/Issues/Risks:** There is the need of a Web Browser to allow a graphical and user-friendly simulation or a application (previously downloaded) to allow simulation from PDAs (just like in the service composition use case)
- **Trigger event(s):** The OPUCE service producer accesses Portal (for Web Editor) or the OPUCE service producer accesses the Mobile Editor application installed on the PDA directly and the user selects the Service Simulation function
- **Primary actor:** OPUCE service producer.
- **Secondary actor(s):** None.
- **Components involved:** Portal, Web Editor, Mobile Editor

4.4.3.2. Use case detail

- **Pathway name:** Service simulation.
- **Trigger event:** The OPUCE service producer, while editing a service, clicks on Service Simulation.
- **Main sequence of steps for Simulation:**
 1. A service composition is open in the Service Editor (Web or Mobile): the user switches to Simulation view
 2. The service composition is shown, but commands for service simulation are available.
 3. The system requests the user to fill-in the values for properties that are needed to start simulation.
 4. The initial event(s) are active on the canvas of the simulation view when the simulation starts.
 5. The user selects the event they want to start with and the simulation starts.
 6. The control flows to the actions connected to the chosen event.
 - The system shows the values of the properties of the service to the user and allows the user to set them
 - The user can set some of the values and confirm
 - The system re-evaluates the depending properties and shows them to the user
 - The system shows the list of events that could be raised at this stage of the service flow
 7. The user selects an event to be raised next, thus going back to the previous step
 8. The simulation stops when the user interrupts it or a leaf of the flow is reached
- **Variations:**
 - *Cross Simulation:* a composition created with Web Editor can be simulated by the Mobile Editor and vice versa.
 - *Internationalization:* a composition can be simulated in any supported locale, regardless of the one used at service creation time; all language-sensitive strings are rendered according to the preferred language of the OPUCE service producer.
- **Extensions:**
 - *Stop and restart:* the user can stop the simulation session and go back to the editing view (e.g. to correct an error that showed up, to enhance the service composition etc.). After changing the service definition, the user can restart the simulation.
- **Business rules:** Service producers will be able to simulate their own services.
- **Special requirements:** Simulation on the Mobile Editor runs on PDAs with limited graphical capabilities, resources (memory, processor, storage, etc.), meanwhile Web Editor runs on desktop browsers with advanced graphical capabilities.
- **Constraints/Issues/Risks:** Simulation runs on the same environment than edition, so a web browser supporting javascript and Ajax is needed for the “Web” version, while a PDA with the Simulation tool installed is needed for the Mobile Editor.

4.4.4. Service Sharing

4.4.4.1. *Use case description information*

- **Name:** Service Sharing
- **Goal:** After service composers finish the creation and deployment of their own services within the service provider's platform, they can share/recommend these services to one or more elements from their social network, which would be considered as the process of service sharing.
- **Use case team leader/members:** ERI / TIL, HUA.
- **Pre-condition:** The service which has been created and deployed by the service composer is running. The service composer is logged in into OPUCE User Portal and they are using some service management portlet (or widget) exhibiting a "service sharing" button. The service composer's OPUCE Social Network is not empty.
- **Post-condition:** The selected end-users receive recommendations to use the shared service.
- **Constraints/Issues/Risks:** Recommendations should not be sent to end-users that are already subscribers to the shared Service. To avoid spam, recommendations should only be allowed to users from service composer's Social Network with an authorisation rule allowing recommendations from the Advertiser End-user.
- **Trigger event(s):** The service composer pushes the "service sharing" button.
- **Primary actor:** Service producer.
- **Secondary actor(s):** End-users receiving recommendations to subscribe the shared service.
- **Components involved:** Portal, UPR, Service Advertiser.

4.4.4.2. *Use case detail*

- **Pathway name:** Service Sharing.
- **Trigger event:** The service composer pushes the "service sharing" button in the user portal.
- **Main sequence of steps:**
 1. The service composer presses the "service sharing" button from the Service UI in order to share a service with their social network.
 2. A pop-up window opens with the user's Social Network list (Social Network), which has been retrieved from UPR, from where the service composer may select one or more elements (or groups).
 3. Optionally the service composer may write a recommendation sentence or pick-up a pre-defined recommendation sentence.
 4. The corresponding module in the platform (Service Advertiser) checks then which of these users do not have the service already subscribed and have authorisation rules allowing advertisements from the service composer;
 5. The Service Advertiser notifies the users about the shared service, identifying the service composer and showing the recommendation sentence.

6. Some options must be available:
 - *Accept and subscribe*: Advertised end-user is directed to the service subscription UI
 - *Reject*: The service advertiser will notify the service composer that the shared service has been reject, with a sentence written by the advertised end-user to show the reason to reject (can be blank).
 - *Reject and Add service composer to Blocked Advertisements list*: Authorisation rule blocking service advertising from the service composer is set.
 - *Cancel*: No further action is performed.
- **Variations**: None.
- **Extensions**: The service composer does not have any contacts in their OPUCE Social Network. The “Service sharing” buttons are all unavailable.
- **Business rules**: None.
- **Special requirements**: Some kind of Service Share Authorisation Rules editor is needed in OPUCE Portal.

4.5. Base service producer Use Cases

Next sections show the description of the use cases that are executed by the role of base service producers, which will mainly be third parties and operator companies.

4.5.1. Base Service creation

This use case explains how to create the service facets for the new service.

4.5.1.1. Use case description information

- **Name**: Base service creation.
- **Goal**: To create and save an OPUCE base service, starting from a set of software modules previously created by an external third party. And to support the base service producer to edit component description (both master and facets description) with a friendly user interface.
- **Use case team leader/members**: UPM / NEC.
- **Pre-condition**: The base service producer has created software modules that will constitute the component. The base service producer has been granted with privileges to access the component editor (i.e. it is a trusted 3rd party). The Base Service producer has been authenticated into the OPUCE platform.
- **Post-condition**: A new base service is created and once deployed it is stored in the component repository, as it will be then when the base service will be available to composition.
- **Constraints/Issues/Risks**: None.

- **Trigger event(s):** The base service producer accesses the component editor.
- **Primary actor:** Base service producer.
- **Secondary actor(s):** None.
- **Components involved:** Component editor, component repository.

4.5.1.2. **Use case detail**

- **Pathway name:** Base service creation.
- **Trigger event:** The base service producer accesses the component editor.
- **Main sequence of steps:**
 1. Base service producer instructs CE to create a new base service.
 2. Base service producer provides the CE with the binary software modules the new OPUCE base service will consist of (e.g., JSLEE SBBs, Java EE apps).
 3. Base service producer provides the CE with the master service description. The CE should provide a friendly user interface to the base service producer in order to populate service description file.
 4. Likewise, the base service producer provides service facets descriptions (in case they are needed) to CE. The CE should provide friendly user interfaces for editing facets.
 5. Base service producer commands the CE to save the base service.
 6. CE checks consistency of the just created base service.
 7. In case validation succeeds, the base service creator might decide to deploy the base service and once deployed it is saved in the component repository.
- **Variations:**
 - *Existing base service edition:* base service producers may choose to edit an existing base service they had previously created. In that case, step 1 is replaced with a selection of an existing base service to be edited and its existence becomes a pre-condition (as well as base service producer authorship).
- **Extensions:**
 - *Invalid service:* if step 6 fails, user is prompted to solve the problems.
 - *User leaving CE:* if base service producer decides to leave CE too early, they may be given the option to discard changes made to the component or save an interim version to be completed later.
- **Business rules:** 3rd parties must have signed an offline SLA to be registered as those; access to underlying operator capabilities by third parties must be studied.
- **Special requirements:** CE must be implemented as a desktop-PC graphical application.
- **Constraints/Issues/Risks:** None.

4.5.2. Base Service management

This use case consists of deployment, activation, withdrawal and monitoring of base services.

4.5.2.1. *Use case description information*

- **Name:** Base service Management.
- **Goal:** Deploy, Monitor and Un-deploy a created base service.
- **Use case team leader/members:** UPM / NEC.
- **Pre-condition:** Base service producer has created a component in the local computer; base service producer has been granted with privileges to access component editor (i.e. it is a trusted 3rd party). The Base Service producer has been authenticated into the OPUCE platform.
- **Post-condition:** A base service is un-deployed on the OPUCE platform.
- **Constraints/Issues/Risks:** None.
- **Trigger event(s):** The base service producer accesses component editor.
- **Primary actor:** Base service producer.
- **Secondary actor(s):** None.
- **Components involved:** Component editor, component repository.

4.5.2.2. *Use case detail*

- **Pathway name:** Base service management.
- **Trigger event:** Base service producer accesses component editor.
- **Main sequence of steps:**
 1. Base service producer opens a previously created base service in their computer.
 2. It validates the base service by clicking in the CE validation feature. It returns a list of errors related to some actions, events and properties not implemented in the Base service execution endpoint.
 3. The producer will correct the errors in their base service package IDE (Integrated Development Environment).
 4. The user tries to validate again the Base service and now it returns success.
 5. The base service creator now deploys the component and uploads the descriptors to the component repository making the base service visible to the OPUCE platform. Finally when the user open's the base service palette in their editor (Mobile or Web), they will see their new component ready for being used in composition.
 6. Now the user can monitor the base service and check if it is being used in any service composition or how many users are currently running Base service instances.

7. After some time using their service the user is not happy with it and wants to remove the base service from the OPUCE platform.
 8. The user then accesses the component editor again
 9. From the platform list of available base service the user selected their previously deployed service and clicks un-deploy on the CE. The platform will check if the base service is running or used in any composition.
 10. The base service is running so the user is warned that the base service is marked for deletion and it is not allowed to be used on any new compositions. When the base service is no more used in any composition it is actually removed from the platform.
 11. Now the base service is no more listed in the base services palette.
 12. The user then monitors the base service and the user can see that the base service is in deletion process.
- **Variations:**
 - Updating a Base service:
 - The producer opens an existing base service in the platform (deployed), in that case the step 1 is replaced by opening a base service from the platform base service palette.
 - When the producer performs step 2 to 5 the base service descriptors and the base service package are verified to check if this version of the component is not already in the platform. Also the old version recent attribute (on the base service main descriptor) is updated to false as new version is now being deployed.
 - **Extensions:**
 - *Invalid service:* if step 6 fails, user is prompted to solve the problems.
 - *User leaving CE:* if base service producer decides to leave CE too early, she may be given the option to discard changes made to the component or save an interim version to be completed later.
 - **Business rules:** 3rd parties must have signed an offline SLA to be registered as those; access to underlying operator capabilities by third parties must be studied.
 - **Special requirements:** CE must be implemented as a desktop-PC graphical application.
 - **Constraints/Issues/Risks:** None.

5. Requirements

5.1. Guidelines

In the subsequent sections several words are used to signify the requirements in the specification. These words are often capitalized and follow the same rules than the IETF documents.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in IETF BCP 14 RFC 2119 [19].

5.2. Service look-up requirements

The following list of requirements is related to the case when the service consumer searches inside the portal for a specific existing service.

- R1. The OPUCE consumer **MUST** be authenticated before accessing the platform to look up any service.
- R2. The platform **MUST** provide an interface that shows the list of existing OPUCE services, sorted by date and alphabetically, and allow the user to search through the list.
- R3. The platform **MUST** allow the user to execute the search by introducing in the corresponding field one or more words related to the kind of service the user wants to find.
- R4. The platform **MUST** respond the user with a list of services related to the word they previously introduced (if existing).
- R5. In case such service needed by the user does not exist, then the platform **MUST** show no services after executing the search.
- R6. Once the search has been executed, and before showing the list of resulting services, the platform **MUST** filter those that the user is not authorised to use.
- R7. Once provided with the list of desired services, the user **MUST** be able to access the services related information, such as service description, to get the exact idea of what each service does and decide whether to subscribe to these services or not.
- R8. The platform **MUST** provide the user with the right interface to subscribe to the service or services the user finds interesting.

5.3. Service Subscription, Personalization and Usage requirements

The personalization feature refers to the possibility of customization of services and components based on personal needs and user profile.

- R9. The OPUCE consumer **MUST** be authenticated before accessing to OPUCE platform to subscribe and execute any service.

- R10. The service **MUST** be already created and deployed in the platform so that the user can subscribe and execute.
- R11. The platform **MUST** provide an interface for the consumers to subscribe to services they are authorised to use.
- R12. The platform **MUST** provide a data base of user profiles and the adequate interfaces for the platform modules to access and retrieve/store such information.
- R13. The platform **MUST** provide an interface to help the service creator to define configurable parameters of the service.
- The service creator **MAY** define parameters that can be provided by the end-user and this will imply different behaviours of the service based on the end-user preferences.
 - The service creator **MAY** establish parameters that can not be modified by the end-user. Only the service creator can modify these parameters when using the service as a base service.
 - The service creator **MAY** establish parameters that will be completed using the user profile like phone preference, etc. and user can not modify it.
- R14. When the user is subscribed, the platform **MUST** check the service description to identify the open properties to be defined before execution.
- The platform **SHOULD** be able to retrieve those parameters which are stored in the user profile database.
 - The platform **MUST** provide a link to an interface to define configurable parameters defined by the user creator for the consumer to fill in at subscription time.
- R15. The platform **MUST** ask new OPUCE subscriber about some basic data (name, mobile phone number, address, language, etc) when registering.
- R16. Once the subscriber has introduced the open properties, the platform **MUST** activate the subscribed service.
- R17. The platform **SHOULD** store in the user profile, for each service, the parameters introduced by the subscriber. These parameters **MAY** be considered by default in subsequent usages of the service.
- R18. If all options for execution are not completed, the platform **MUST** ask for the user preferences which were not introduced at subscription time.
- In some cases, these preferences **MAY** be established in an automatic way, without asking the user (through the user profile for this service established by the user after registration). The user **MAY** modify and complete them.
- R19. The platform **MUST** allow changing the open properties that were introduced at subscription time.
- R20. The platform **MUST** provide an interface for the consumer to run the subscribed services.
- R21. The service **SHOULD** be adapted to user context during execution.

5.4. Direct Service Advertising Requirements

These are the requirements extracted from the case when a user of the OPUCE platform, a service consumer, finds a service that wants to recommend to their contacts belonging to their social network.

- R22. OPUCE consumer **MUST** be authenticated before recommending services to other users.
- R23. Every sharable Service **MUST** have “advertise this” button available from the Subscribed User Portal
- R24. The platform **MUST** provide social network capabilities
- R25. The advertiser end-user **MUST** be able to select one or more elements (or groups) from their Social Network list.
- R26. The advertiser end-user **MAY** write a recommendation sentence or pick-up a pre-defined recommendation sentence
- R27. The platform **SHOULD NOT** notify users that have already subscribed the service
- R28. The platform **MUST** ensure that notified users have authorisation rules allowing advertisements from the advertiser end-user
- R29. The service advertising notification **SHOULD** identify the Advertiser end-user and show the recommendation sentence, when available.
- R30. The advertised end-user **SHOULD** be able to accept the recommended service and be directed to the service subscription user interface, from the notification itself
- R31. The advertised end-user **SHOULD** be able to reject from the notification itself
- R32. The advertised end-user **SHOULD** be able to add advertiser to Blocked Advertisements list, setting an authorisation rule that blocks service advertising from Advertiser end-user
- R33. The User Portal **SHOULD** provide a Service Share Authorisation Rules editor
- R34. The Service Share Authorisation rules **MUST** be handled by the User Profile Repository

5.5. Service Recommendation Requirements

These are the requirements extracted from recommendation use case, which is the feature of the platform that enables a user to be notified when any desired or useful service for them is founded.

- R35. The platform **MUST** provide an interface to help the user to add/modify preferences in their profile.
- R36. Sensors **MAY** collect and store all the usage information in the platform.
- R37. Information collected by sensors **SHOULD** be processed by a learning process to guess user preferences.
- R38. The learning process **SHOULD** access the user profile to store the preferences inferred by monitoring the Service Usage (and using some algorithm on them)

- R39. The Platform SHOULD provide an interface to show the preferences inferred to the user.
- R40. The platform SHOULD provide an interface to allow users to accept/refuse the inferred preferences by the learning process
- R41. The platform MUST provide the way to identify new services to recommend the users when Service preferences of the user profile changes; match user-service
- R42. All elements of the service SHOULD have a categorization or a Semantic description.
- R43. Preferences SHOULD be related with the descriptions of each element that composes the services. All this concept SHOULD belong to the same semantic source (taxonomy/ontology)
- R44. The Platform SHOULD provide an interface to add a semantic description of each service and base services that compose the services.
- R45. The Platform MAY provide an interface to start directly the recommendation process.
- R46. The User Profile Repository MUST trigger the notification mechanism (to trigger the Matcher Broker). This is triggered every time a profile changes. It means a new one is added or an old one is updated
- R47. The Matcher Broker MUST provide an interface to receive notification from the User Profile Repository when a profile changes.
- R48. The Matcher Broker MUST have access to both Service and user profile repositories.
- R49. The Matcher Broker MUST select the right Matcher Provider from the information available. More than one Matcher Providers MAY be available.
- R50. The Matcher Broker MAY merge the results of more than one Matcher Providers used by the Recommendation process.
- R51. The platform MUST notify about recommended services to the user in the preferred way.

5.6. Remote Service Management requirements

Remote Service Management is the mechanism inside the platform that automatically manages the deployment of remote base services during the phase of a service subscription by an end user. When the user subscribes for a service, the RSM collects all the service facets in which the required information is stored. In case of remote base services, it deploys them on the user end device. The requirements related to RSM use case are:

- R52. Some Base Services MAY need remote deployment.
- R53. The user MUST have the Remote Service Management widget installed on the respective terminal
- R54. The Deployment Manager submodule RSM MUST have access to all the resources of the service subscribed (facets file).
- R55. The RSM widget MUST implement an SLM interface to receive the command to deploy and arm new elements needed for the new service.

- R56. The Deployment Manager submodule RSM MUST access the UPR to retrieve the User profile.
- R57. The user MUST be registered on the Platform and MUST have a SIP account.
- R58. The RSM Widget MUST be registered at a SIP registrar and be reachable by the OPUCE platform.
- R59. The Deployment Manager submodule RSM MUST be registered at a SIP registrar and be reachable by the user terminals.
- R60. The Deployment Manager submodule RSM MUST send SIP messages to the user containing the description of each remote base service to be installed, as well as the way to get them.
- R61. The RSM widget on the platform MUST show a message to inform on the new installation.
- R62. The RSM widget MUST download and install the base services which are necessary.
- R63. The user SHOULD be able to refuse to install any elements on their terminal.
- R64. The Deployment Manager submodule RSM MUST implement an interface to receive the status of the installed base services on each subsystem; especially on the terminal of the user.
- R65. If something goes wrong the Deployment Manager submodule RSM SHOULD implement a rollback mechanism to clean the deployment procedure.
- R66. The Deployment Manager submodule RSM MUST return the state after the deployment procedure to allow the start of a service after successful deployment.

5.7. Context Usage and Feed requirements

Context and usage feed is the client on the user device responsible for collecting all the information related to the user behaviour and all the features and capabilities of the device. All this information is then sent to the OPUCE platform to be analysed, processed and then made available to the personalization process. The list of requirements dealing with this use case is:

- R67. The platform MUST provide a set of Context Usage and Feed (CUF) clients to retrieve context information from clients dynamically. Such clients will collect all the available row data from the device.
- R68. One CUF client MUST be installed on the user device to provide context feed functionalities.
- R69. The CUF client MUST provide an interface to help the user to configure their credentials.
- R70. The CUF client MUST provide an interface to help the user to allow/deny the collection of any information about them.
- R71. The CUF client MUST envelop the data in a XML document.
- R72. The CUF client MUST have access to the OPUCE platform to send the collected data.
- R73. The CUF client MUST send the document with the context information to the platform to be stored.

- R74. The platform **MUST** provide a repository where Context information should be stored. Such repository **MUST** implement an interface to get the data from the client side.
- R75. The user **MAY** stop to send their own data to the Platform in any moment.

5.8. Service Composition Requirements

Following are the list of requirements extracted from the service creation related activities. The service creation is the process in which complex services are composed by combining atomic base services.

- R76. There **MUST** be a Web Editor, which allows a graphical and user-friendly composition with only a web browser.
- R77. There **MUST** be a Mobile Editor to allow composition from mobile devices like PDAs.
- R78. The interface of both Web and Mobile Editors **MUST** be user-friendly to allow the service producer to create or modify a service composition and to design the service flow easily.
- R79. The service producer needs an extensible set of base services to compose their services; this set **MUST** be authorized for the registered user.
- R80. The Base Services available for service composition **MUST** be split in categories, to ease their retrieval.
- R81. The Editors **SHOULD** provide functionalities to search the Base Services by keywords.
- R82. The platform **SHOULD** store the Base Services jointly with the keywords/tags associated. Semantic reasoning **SHOULD** exist.
- R83. There **MUST** be the possibility to delete base services already inserted in a composition.
- R84. Connections between base services **MUST** be able to be deleted or changed.
- R85. There **MUST** be a simple way to set the properties for each block of the composition.
- R86. In the property configuration, the composer **MAY** insert references to values from other blocks in the same composition.
- R87. In the property configuration, the composer **MAY** insert references to values from the user profile (\$me).
- R88. The Editors **SHOULD** assist the service composition process as much as possible, for example performing automated linking between actions and events, whenever possible.
- R89. The service producer **MAY** change any default value or automatic links suggested by the Editors.
- R90. A composition created with Web Editor **MUST** be able to be edited by the Mobile Editor and vice versa, a composition created by the Mobile Editor **MUST** be able to be edited also in the Web Editor.
- R91. The service editors **MUST** support internationalization of the functions and menu shown.

- R92. The composition itself **MUST** be presented in the language chosen by the composer, translating each visible string in the service composition into the selected language.
- R93. If some base service element does not have its name translated in the composer's chosen language, a default value **MUST** be displayed.
- R94. A composition **MUST** be edited in any supported locale, regardless of the one used at service creation time.
- R95. The editors **MUST** give the possibility to the service producer to save the service composition in the repository.
- R96. The Mobile Editor **MUST** let the user to save locally the service composition.
- R97. The editors **MUST** warn the service producer if errors in the composition are found.
- R98. The editors **MUST** permit existing services to be opened and re-edited.

5.9. Service management requirements

Service management represents the actions that service creators execute in order to manage services created by them, actions such as deployment, provisioning, activation, scheduling, withdrawal and monitoring. The requirements derived from Service Management use case are:

- R99. The platform **MUST** support the ability to perform automated deployment (including configuration, installation, activation, publishing) and removal/withdrawal of components, applications and services.
- R100. The platform **SHOULD** provide a mechanism to ensure that other services or base services will be considered when a change of state (e.g. installation or removal) of a base service happens.
- R101. Each base service in the platform **SHOULD** support the capability to allow the service provider and end user to perform life cycle functions, such as install, upgrade, downgrade, stop and start of the base service.
- R102. It **MUST** be possible to perform life cycle functions on execution instances of base services.
- R103. The platform **MUST** provide a mechanism to support the configuration, registration and activation (start and stop) of an OPUCE service or base service.
- R104. The platform **MUST** provide to the service provider and end user all information associated to the existence of services and base services, and their relationship.
- R105. The platform **SHOULD** provide a mechanism that notifies the state change of either a service or a base service, such as service evolution, withdrawal, etc
- R106. The platform **MUST** provide a user interface for the configuration, registration, publication and activation (start and stop) of a service and base service.
- R107. The OPUCE platform **MUST** provide a mechanism to manage data related with the user's subscriptions to services.
- R108. Each base service or service **SHOULD** expose through a standard interface their characteristics, which includes:

- Service Logic
- Supported Interfaces
- Configuration data
- Deployment (including provisioning) tasks
- Semantic data

5.10. Service Simulation requirements

Service Simulation is the chance, given to service producers, to simulate locally the behaviour of a service flow they have designed.

- R109. The simulator **MUST** be able to open a GSD created by the Mobile or Web Editor.
- R110. The simulator is a debug tool that **MUST** support standalone verification of a service composition.
- R111. A service simulation function **SHOULD** be accessible from all the Service Editors: Web and Mobile.
- R112. A composition created with Web Editor **MAY** be simulated by the Mobile Editor and vice versa.
- R113. Simulator **MUST** have a user-friendly interface to allow the service producer to set properties and to control the flow.
- R114. Simulator **MUST** ask to introduce the values for properties that are needed to start simulation.
- R115. If the service contains \$me references they **MUST** be resolved with the user profile data of the service producer.
- R116. The simulator **MUST** provide feedbacks about possible problems found during the execution of the service definition.
- R117. Simulator **SHOULD NOT** be integrated with SEE so that service behaviour can be simulated.
- R118. Simulation **MUST** be event driven.
- R119. Simulation **MUST** be interactive.
- R120. Simulation **MUST** go on step-by-step asking all the needed values to the service producer.
- R121. The simulator **MUST** provide control of event triggering to the service producer.
- R122. The service producer **MUST** be allowed to stop the simulation session.
- R123. The service producer **MUST** be allowed to go back to the editing view.
- R124. The simulation **MUST** be adapted to the language selected by the service creator.

5.11. Service Sharing requirements

After service composers finish creating and deploying their own services within the service provider's platform, they can recommend these services to one or more elements from their social network, which would be considered as the process of service sharing.

- R125. The portal **MUST** provide an edit environment for the service composer to specify the target people they want to advertise.
- R126. The edit environment **MAY** let service composer write a comment or sentence regarding the service before sending the service sharing request.
- R127. The edit environment **SHOULD** allow the service composer to select one or more elements (or groups) from their Social Network.
- R128. The service advertiser **MUST** provide an interface to receive the service sharing request from the edit environment which is initiated by the service composer.
- R129. After receiving a service sharing request, the service advertiser **SHOULD** check if the target people involved in the service sharing request have subscribed this service.
- R130. The platform **MUST** provide different methods for the target people to receive the service sharing message.
- R131. The platform **SHOULD** provide an interface for the target people to revise preferences to specify how to receive the service sharing message.
- R132. The platform **SHOULD** provide four options for the targeted people to choose how to reply to the service sharing message, that is: Accept and subscribe, Reject, Reject and Add the service composer to blocked list, Cancel (no further action is performed).

5.12. Base Service Creation requirements

These are the requirements to create the service facets for the new base service.

- R133. The OPUCE platform **MUST** provide 3rd parties with facilities to make externally created OPUCE components available in the platform. These facilities are provided by a software application called Component Editor.
- R134. The OPUCE Component Editor **MUST** build up OPUCE components from JSLEE components or J2EE components, provided by the Component Editor user.
- R135. The OPUCE Component Editor **MUST** be responsible to make new components ready into the OPUCE platform.
- R136. The OPUCE Component Editor **MAY** allow the user to work offline.
- R137. The OPUCE Component Editor **MUST** check user authentication and authorization privileges in case that any online interaction with the OPUCE platform is performed.
- R138. The OPUCE Component Editor GUI **MUST** provide component access repository features (open, save, deploy, undeploy and delete) and **MAY** create a new component from scratch.

- R139. The OPUCE Component Editor MUST provide a graphical user interface to help in component creation. It will allow to edit the master document and the different component facets.
- R140. The OPUCE Component Editor GUI MUST provide browsing of all the required objects for the component build-up process.
- R141. The OPUCE Component Editor MUST allow to edit all the facets needed to define a component. It MUST provide at least a fallback, generic edition UI to edit any facet. It MAY also provide UI elements tailored to edit each facet.
- R142. The OPUCE Component Editor MUST allow to input and edit general identification data, and the identification of each of the components facets, which are stored in separate files. It will allow editing the following fields: ComponentID, Name, Version, Owner.
- R143. The OPUCE Component Editor MUST allow to define, for all the facets, at least ComponentID, language and type to identify the facet itself.
- R144. The OPUCE Component Editor MUST allow to define one list of other components required to run the component being defined, for the provisioning facet.
- R145. The OPUCE Component Editor MUST provide definition of the execution environment required by the component, for the provisioning facet.
- R146. The OPUCE Component Editor MAY allow defining a set of keywords that will be used by the SCE to find semantic compatible functions, for the semantic facet.
- R147. The OPUCE Component Editor MUST allow to define the function, properties and events that the component exposes to the external world, for the Interface facet. It consists of different levels: first the nomenclature of the interface, that can be used by the service composer to find appropriate linkage between components and second the WSDL facet, that exposes the functional end-points in the service execution environment.
- R148. The OPUCE Component Editor MUST allow to define a list of initial events required to start the component, as well as a list of final events produced by the component, so that the resources can be freed after these events.
- R149. The OPUCE Component Editor MUST allow to define a set or rules regarding the calling order of the component functions, for the behaviour constraints facet.
- R150. The OPUCE Component Editor MUST allow to define the requirements at platform/network level of the component, for the Service level agreement facet. This will include the limit number of concurrent instances of the component, the used bandwidth, etc.
- R151. The OPUCE Component Editor MUST provide mechanisms to supply the mandatory icons supported by the service editors.
- R152. The information regarding the icons associated with the new base service SHOULD be provided within the service description.
- R153. The OPUCE Component Editor MUST allow saving component descriptors on user local computer.
- R154. The OPUCE Component Editor MUST upload component executable package to the SLM/Deployment Manager and store the descriptors in the SLM/Repository.
- R155. The OPUCE Component Editor MUST check the completeness of the uploaded OPUCE Component before proceeding to upload executable package and store descriptors.

- R156. The OPUCE Component Editor **MUST** validate that all dependencies are satisfied before readying a component onto the platform.
- R157. The OPUCE Component Editor **MAY** interact with a component, after being deployed in their container, to testing its functionality before storing the descriptors in the repository, that will then give visibility to the base service in the platform. For that a component test feature provided by the OPUCE platform may be used.
- R158. The OPUCE Component Editor **SHOULD** support different user languages.

5.13. Base Service Management requirements

In this case, following requirements correspond to the deployment, activation, withdrawal and monitoring of a base service.

- R159. The component Editor **MUST** provide a Hall page that lists the components, local and from the repository, and allows the user to open one of them.
- R160. The component executable package **MUST** be provided by the Component Editor user (not provided by the OPUCE platform) and the component editor will be responsible for deploying it into the OPUCE platform.
- R161. 3rd parties **MUST** be given the possibility to authenticate into OPUCE platform.
- R162. 3rd parties **MUST** have the option to authenticate in the platform or to work offline.
- R163. The component descriptors **MUST** be saved on the user local computer together with a Component Editor Project file, in a separate folder for each Component Editor.
- R164. The component editor **MUST** provide a button to validate the component by:
- Checking if events generated by actions exist on the event list
 - Validating generated facets against provided schemas
 - Checking if the properties referred by the actions or events exist
- R165. The Component Editor **MUST** provide a button or menu item to deploy the component in the OPUCE platform. The component executable package will be uploaded to the SLM/Deployment Manager and the descriptors will be stored in the SLM/Repository.
- R166. The OPUCE platform **MUST** provide a 3rd party Web service interface to interact with the component editor.
- R167. From the platform 3rd party interface web service, the Component Editor **MUST** be able to access the SLM (Repository/Deployment Manager).
- R168. From the platform 3rd party interface web service, the Component Editor **MUST** be able to access the User Information Management.
- R169. The editor **MUST** be linked to the AAA interface to get authentication and authorization into the platform 3rd party web service interface.
- R170. The editor **MUST** provide a page that lists:
- How many service compositions are referring to this Component
 - If the component is being run and how many instances exist

- R171. The editor **MUST** allow to undeploy components created by the current user.
- R172. The component editor **SHOULD** not allow to update a component with a version that already exists in the repository.
- R173. If the user closes the editor without saving, it **MUST** be prompted to save, discard or exit the changes made.
- R174. If the user tries to update an existing component in the repository, the component editor **MUST** create a new component with the updated version.

5.14. Platform modules classification of requirements

The following table shows a classification of the previous requirements grouped by platform modules. This classification will help related tasks to implement the corresponding modules by fulfilling these requirements.

Portal	<p><i>Service look-up</i></p> <p>R2. The platform MUST provide an interface that shows the list of existing OPUCE services, sorted by date and alphabetically, and allow the user to search through the list.</p> <p>R3. The platform MUST allow the user to execute the search by introducing in the corresponding field one or more words related to the kind of service the user wants to find.</p> <p>R7. Once provided with the list of desired services, the user MUST be able to access the services related information, such as service description, to get the exact idea of what each service does and decide whether to subscribe to these services or not.</p> <p>R8. The platform MUST provide the user with the right interface to subscribe to the service or services the user finds interesting.</p> <p><i>Service subscription, personalization and usage</i></p> <p>R11. The platform MUST provide an interface for the consumers to subscribe to services they are authorised to use.</p> <p>R13. The platform MUST provide an interface to help the service creator to define configurable parameters of the service.</p> <p>R20. The platform MUST provide an interface for the consumer to run the subscribed services.</p> <p><i>Direct Service Advertising</i></p> <p>R23. Every sharable Service MUST have “advertise this” button available from the Subscribed User Portal</p> <p>R25. The advertiser end-user MUST be able to select one or more elements (or groups) from their Social Network list.</p> <p>R26. The advertiser end-user MAY write a recommendation sentence or pick-up a pre-defined recommendation sentence</p> <p>R33. The User Portal SHOULD provide a Service Share Authorisation Rules editor</p> <p><i>Service Recommendation</i></p>
---------------	---

R39. The Platform SHOULD provide an interface to show the preferences inferred to the user.

R40. The platform SHOULD provide an interface to allow users to accept/refuse the inferred preferences by the learning process.

R44. The Platform SHOULD provide an interface to add a semantic description of each service and base services that compose the services.

R45. The Platform MAY provide an interface to start directly the recommendation process.

Service Composition

R76. There MUST be a Web Editor, which allows a graphical and user-friendly composition with only a web browser.

R77. There MUST be a Mobile Editor to allow composition from mobile devices like PDAs.

R78. The interface of both Web and Mobile Editors MUST be user-friendly to allow the service producer to create or modify a service composition and to design the service flow easily.

R80. The Base Services available for service composition MUST be split in categories, to ease their retrieval.

R81. The Editors SHOULD provide functionalities to search the Base Services by keywords.

R83. There MUST be the possibility to delete base services already inserted in a composition.

R84. Connections between base services MUST be able to be deleted or changed.

R85. There MUST be a simple way to set the properties for each block of the composition.

R86. In the property configuration, the composer MAY insert references to values from other blocks in the same composition.

R87. In the property configuration, the composer MAY insert references to values from the user profile (\$me).

R88. The Editors SHOULD assist the service composition process as much as possible, for example performing automated linking between actions and events, whenever possible.

R89. The service producer MAY change any default value or automatic links suggested by the Editors.

R90. A composition created with Web Editor MUST be able to be edited by the Mobile Editor and vice versa, a composition created by the Mobile Editor MUST be able to be edited also in the Web Editor.

R91. The service editors MUST support internationalization of the functions and menu shown.

R92. The composition itself MUST be presented in the language chosen by the composer, translating each visible string in the service composition into the selected language.

R93. If some base service element does not have its name translated in the composer's chosen language, a default value MUST be displayed.

R94. A composition MUST be edited in any supported locale, regardless of the one used at service creation time.

R95. The editors MUST give the possibility to the service producer to save the service composition in the repository.

- R96.** The Mobile Editor **MUST** let the user to save locally the service composition.
- R97.** The editors **MUST** warn the service producer if errors in the composition are found.
- R98.** The editors **MUST** permit existing services to be opened and re-edited.

Service Management

- R104.** The platform **MUST** provide to the service provider and end user all information associated to the existence of services and base services, and their relationship.
- R106.** The platform **MUST** provide a user interface for the configuration, registration, publication and activation (start and stop) of a service and base service.

Service Simulation

- R109.** The simulator **MUST** be able to open a GSD created by the Mobile or Web Editor.
- R110.** The simulator is a debug tool that **MUST** support standalone verification of a service composition.
- R111.** A service simulation function **SHOULD** be accessible from all the Service Editors: Web and Mobile. Web and Mobile.
- R112.** A composition created with Web Editor **MAY** be simulated by the Mobile Editor and vice versa.
- R113.** Simulator **MUST** have a user-friendly interface to allow the service producer to set properties and to control the flow.
- R114.** Simulator **MUST** ask to introduce the values for properties that are needed to start simulation.
- R116.** The simulator **MUST** provide feedbacks about possible problems found during the execution of the service definition.
- R118.** Simulation **MUST** be event driven.
- R119.** Simulation **MUST** be interactive.
- R120.** Simulation **MUST** go on step-by-step asking all the needed values to the service producer.
- R121.** The simulator **MUST** provide control of event triggering to the service producer.
- R122.** The service producer **MUST** be allowed to stop the simulation session.
- R123.** The service producer **MUST** be allowed to go back to the editing view.
- R124.** The simulation **MUST** be adapted to the language selected by the service creator.

Service Sharing

- R125.** The portal **MUST** provide an edit environment for the service composer to specify the target people they want to advertise.
- R126.** The edit environment **MAY** let service composer write a comment or sentence regarding the service before sending the service sharing request.
- R127.** The edit environment **SHOULD** allow the service composer to select one or more elements (or groups) from their Social Network.
- R131.** The platform **SHOULD** provide an interface for the target people to revise preferences to specify how to receive the service sharing message.
- R132.** The platform **SHOULD** provide four options for the targeted people to

choose how to reply to the service sharing message, that is: Accept and subscribe, Reject, Reject and Add the service composer to blocked list, Cancel (no further action is performed).

Base Service Creation

R133. The OPUCE platform **MUST** provide 3rd parties with facilities to make externally created OPUCE components available in the platform. These facilities are provided by a software application called Component Editor.

R134. The OPUCE Component Editor **MUST** build up OPUCE components from JSLEE components or J2EE components, provided by the Component Editor user.

The OPUCE Component Editor MUST be responsible to make new components ready into the OPUCE platform.

R136. The OPUCE Component Editor **MAY** allow the user to work offline.

R138. The OPUCE Component Editor GUI **MUST** provide component access repository features (open, save, deploy, undeploy and delete) and **MAY** create a new component from scratch.

The OPUCE Component Editor **MUST** provide a graphical user interface to help in component creation. It will allow to edit the master document and the different component facets.

R140. The OPUCE Component Editor GUI **MUST** provide browsing of all the required objects for the component build-up process.

R141. The OPUCE Component Editor **MUST** allow to edit all the facets needed to define a component. It **MUST** provide at least a fallback, generic edition UI to edit any facet. It **MAY** also provide UI elements tailored to edit each facet.

R142. The OPUCE Component Editor **MUST** allow to input and edit general identification data, and the identification of each of the components facets, which are stored in separate files. It will allow editing the following fields: ComponentID, Name, Version, Owner.

R143. The OPUCE Component Editor **MUST** allow to define, for all the facets, at least ComponentID, language and type to identify the facet itself.

R144. The OPUCE Component Editor **MUST** allow to define one list of other components required to run the component being defined, for the provisioning facet.

R145. The OPUCE Component Editor **MUST** provide definition of the execution environment required by the component, for the provisioning facet.

R146. The OPUCE Component Editor **MAY** allow defining a set of keywords that will be used by the SCE to find semantic compatible functions, for the semantic facet.

R147. The OPUCE Component Editor **MUST** allow to define the function, properties and events that the component exposes to the external world, for the Interface facet. It consists of different levels: first the nomenclature of the interface, that can be used by the service composer to find appropriate linkage between components and second the WSDL facet, that exposes the functional end-points in the service execution environment.

R148. The OPUCE Component Editor **MUST** allow to define a list of initial events required to start the component, as well as a list of final events produced by the component, so that the resources can be freed after these events.

R149. The OPUCE Component Editor **MUST** allow to define a set or rules regarding the calling order of the component functions, for the behaviour constraints facet.

	<p>R150. The OPUCE Component Editor MUST allow to define the requirements at platform/network level of the component, for the Service level agreement facet. This will include the limit number of concurrent instances of the component, the used bandwidth, etc.</p> <p>R151. The OPUCE Component Editor MUST provide mechanisms to supply the mandatory icons supported by the service editors.</p> <p>R153. The OPUCE Component Editor MUST allow saving component descriptors on user local computer.</p> <p>R155. The OPUCE Component Editor MUST check the completeness of the uploaded OPUCE Component before proceeding to upload executable package and store descriptors.</p> <p>R156. The OPUCE Component Editor MUST validate that all dependencies are satisfied before readying a component onto the platform.</p> <p><i>Base Service Management</i></p> <p>R159. The component Editor MUST provide a Hall page that lists the components, local and from the repository, and allows the user to open one of them.</p> <p>R160. The component executable package MUST be provided by the Component Editor user (not provided by the OPUCE platform) and the component editor will be responsible for deploying it into the OPUCE platform.</p> <p>R163. The component descriptors MUST be saved on the user local computer together with a Component Editor Project file, in a separate folder for each Component Editor.</p> <p>R164. The component editor MUST provide a button to validate the component by:</p> <ul style="list-style-type: none"> a. Checking if events generated by actions exist on the event list b. Validating generated facets against provided schemas c. Checking if the properties referred by the actions or events exist <p>R166. The OPUCE platform MUST provide a 3rd party Web service interface to interact with the component editor.</p> <p>R170. The editor MUST provide a page that lists:</p> <ul style="list-style-type: none"> a. How many service compositions are referring to this Component b. If the component is being run and how many instances exist <p>R173. If the user closes the editor without saving, it MUST be prompted to save, discard or exit the changes made.</p>
<p>User Information Management (UIM)</p>	<p><i>User registering (*)</i></p> <p>R15. The platform MUST ask new OPUCE subscriber about some basic data (name, mobile phone number, address, language, etc) when registering.</p> <p><i>Service look-up</i></p> <p>R1. The OPUCE consumer MUST be authenticated before accessing the platform to look up any service.</p> <p>R6. Once the search has been executed, and before showing the list of resulting services, the platform MUST filter those that the user is not authorised to use.</p> <p><i>Service Subscription, Personalization and Usage</i></p>

R9. The OPUCE consumer **MUST** be authenticated before accessing to OPUCE platform to subscribe and execute any service.

R12. The platform **MUST** provide a data base of user profiles and the adequate interfaces for the platform modules to access and retrieve/store such information.

Direct Service Advertising

R22. OPUCE consumer **MUST** be authenticated before recommending services to other users.

R24. The platform **MUST** provide social network capabilities

R28. The platform **MUST** ensure that notified users have authorisation rules allowing advertisements from the advertiser end-user.

R32. The advertised end-user **SHOULD** be able to add advertiser to Blocked Advertisements list, setting an authorisation rule that blocks service advertising from Advertiser end-user.

R34. The Service Share Authorisation rules **MUST** be handled by the User Profile Repository

Service Recommendation

R35. The platform **MUST** provide an interface to help the user to add/modify preferences in their profile.

R37. Information collected by sensors **SHOULD** be processed by a learning process to guess user preferences.

R38. The learning process **SHOULD** access the user profile to store the preferences inferred by monitoring the Service Usage (and using some algorithm on them).

R46. The User Profile Repository **MUST** trigger the notification mechanism (to trigger the Matcher Broker). This is triggered every time a profile changes. It means a new one is added or an old one is updated

R51. The platform **MUST** notify about recommended services to the user in the preferred way.

Service Management

R107. The OPUCE platform **MUST** provide a mechanism to manage data related with the user's subscriptions to services.

Service Simulation

R115. If the service contains \$me references they **MUST** be resolved with the user profile data of the service producer.

Base Service Creation

R137. The OPUCE Component Editor **MUST** check user authentication and authorization privileges in case that any online interaction with the OPUCE platform is performed.

Base Service Management

R161. 3rd parties **MUST** be given the possibility to authenticate into OPUCE platform.

R162. 3rd parties **MUST** have the option to authenticate in the platform or to work

	<p>offline.</p> <p>R168. From the platform 3rd party interface web service, the Component Editor MUST be able to access the User Information Management.</p> <p>R169. The editor MUST be linked to the AAA interface to get authentication and authorization into the platform 3rd party web service interface.</p>
<p>Service Advertising (SA)</p>	<p><i>Direct Service Advertising</i></p> <p>R27. The platform SHOULD NOT notify users that have already subscribed the service</p> <p>R29. The service advertising notification SHOULD identify the Advertiser end-user and show the recommendation sentence, when available.</p> <p><i>Service Recommendation</i></p> <p>R41. The platform MUST provide the way to identify new services to recommend the users when Service preferences of the user profile changes; match user-service</p> <p>R43. Preferences SHOULD be related with the descriptions of each element that composes the services. All this concept SHOULD belong to the same semantic source (taxonomy/ontology)</p> <p>R47. The Matcher Broker MUST provide an interface to receive notification from the User Profile Repository when a profile changes.</p> <p>R48. The Matcher Broker MUST have access to both Service and user profile repositories.</p> <p>R49. The Matcher Broker MUST select the right Matcher Provider from the information available. More than one Matcher Providers MAY be available.</p> <p>R50. The Matcher Broker MAY merge the results of more than one Matcher Providers used by the Recommendation process.</p> <p><i>Context Usage and Feed</i></p> <p>R74. The platform MUST provide a repository where Context information should be stored. Such repository MUST implement an interface to get the data from the client side.</p> <p><i>Service Sharing</i></p> <p>R128. The service advertiser MUST provide an interface to receive the service sharing request from the edit environment which is initiated by the service composer.</p> <p>R129. After receiving a service sharing request, the service advertiser SHOULD check if the target people involved in the service sharing request have subscribed this service.</p> <p>R130. The platform MUST provide different methods for the target people to receive the service sharing message.</p>
<p>Context Awareness (CA)</p>	<p><i>Service subscription, personalization and usage</i></p> <p>R21. The service SHOULD be adapted to user context during execution.</p>

Service Lifecycle Management (SLM)	<p><i>Service look-up</i></p> <p>R4. The platform MUST respond the user with a list of services related to the word they previously introduced (if existing).</p> <p>R5. In case such service needed by the user does not exist, then the platform MUST show no services after executing the search.</p> <p><i>Service subscription, personalization and usage</i></p> <p>R10. The service MUST be already created and deployed in the platform so that the user can subscribe and execute.</p> <p>R14. When the user is subscribed, the platform MUST check the service description to identify the open properties to be defined before execution.</p> <p>R17. The platform SHOULD store in the user profile, for each service, the parameters introduced by the subscriber. These parameters MAY be considered by default in subsequent usages of the service.</p> <p>R18. If all options for execution are not completed, the platform MUST ask for the user preferences which were not introduced at subscription time.</p> <p>R19. The platform MUST allow changing the open properties that were introduced at subscription time.</p> <p><i>Service Recommendation</i></p> <p>R42. All elements of the service SHOULD have a categorization or a Semantic description.</p> <p><i>Remote Service Management</i></p> <p>R52. Some Base Services MAY need remote deployment.</p> <p>R54. The Deployment Manager submodule RSM MUST have access to all the resources of the service subscribed (facets file).</p> <p>R56. The Deployment Manager submodule RSM MUST access the UPR to retrieve the User profile.</p> <p>R59. The Deployment Manager submodule RSM MUST be registered at a SIP registrar and be reachable by the user terminals.</p> <p>R60. The Deployment Manager submodule RSM MUST send SIP messages to the user containing the description of each remote base service to be installed, as well as the way to get them.</p> <p>R64. The Deployment Manager submodule RSM MUST implement an interface to receive the status of the installed base services on each subsystem; especially on the terminal of the user.</p> <p>R65. If something goes wrong the Deployment Manager submodule RSM SHOULD implement a rollback mechanism to clean the deployment procedure.</p> <p>R66. The Deployment Manager submodule RSM MUST return the state after the deployment procedure to allow the start of a service after successful deployment.</p>
---	---

Service Composition

R79. The service producer needs an extensible set of base services to compose their services; this set **MUST** be authorized for the registered user.

R82. The platform **SHOULD** store the Base Services jointly with the keywords/tags associated. Semantic reasoning **SHOULD** exist.

Service Management

R99. The platform **MUST** support the ability to perform automated deployment (including configuration, installation, activation, publishing) and removal/withdrawal of components, applications and services.

R100. The platform **SHOULD** provide a mechanism to ensure that other services or base services will be considered when a change of state (e.g. installation or removal) of a base service happens.

R101. Each base service in the platform **SHOULD** support the capability to allow the service provider and end user to perform life cycle functions, such as install, upgrade, downgrade, stop and start of the base service.

R102. It **MUST** be possible to perform life cycle functions on execution instances of base services.

R103. The platform **MUST** provide a mechanism to support the configuration, registration and activation (start and stop) of an OPUCE service or base service.

R105. The platform **SHOULD** provide a mechanism that notifies the state change of either a service or a base service, such as service evolution, withdrawal, etc

R108. Each base service or service **SHOULD** expose through a standard interface their characteristics, which includes:

Base Service Creation

R152. The information regarding the icons associated with the new base service **SHOULD** be provided within the service description.

R154. The OPUCE Component Editor **MUST** upload component executable package to the SLM/Deployment Manager and store the descriptors in the SLM/Repository.

R157. The OPUCE Component Editor **MAY** interact with a component, after being deployed in their container, to testing its functionality before storing the descriptors in the repository, that will then give visibility to the base service in the platform. For that a component test feature provided by the OPUCE platform may be used.

Base Service Management

R165. The Component Editor **MUST** provide a button or menu item to deploy the component in the OPUCE platform. The component executable package will be uploaded to the SLM/Deployment Manager and the descriptors will be stored in the SLM/Repository.

R167. From the platform 3rd party interface web service, the Component Editor **MUST** be able to access the SLM (Repository/Deployment Manager).

	<p>R171. The editor MUST allow to undeploy components created by the current user.</p> <p>R172. The component editor SHOULD not allow to update a component with a version that already exists in the repository.</p> <p>R174. If the user tries to update an existing component in the repository, the component editor MUST create a new component with the updated version.</p>
Service Execution (SEE)	<p><i>Service Simulation</i></p> <p>R117. Simulator SHOULD NOT be integrated with SEE so that service behaviour can be simulated.</p>
End user device	<p><i>Direct Service Advertising</i></p> <p>R30. The advertised end-user SHOULD be able to accept the recommended service and be directed to the service subscription user interface, from the notification itself.</p> <p>R31. The advertised end-user SHOULD be able to reject from the notification itself.</p> <p><i>Service Recommendation</i></p> <p>R36. Sensors MAY collect and store all the usage information in the platform.</p> <p><i>Remote Service Management</i></p> <p>R53. The user MUST have the Remote Service Management widget installed on the respective terminal</p> <p>R55. The RSM widget MUST implement an SLM interface to receive the command to deploy and arm new elements needed for the new service.</p> <p>R58. The RSM Widget MUST be registered at a SIP registrar and be reachable by the OPUCE platform.</p> <p>R61. The RSM widget on the platform MUST show a message to inform on the new installation.</p> <p>R62. The RSM widget MUST download and install the base services which are necessary.</p> <p>R63. The user SHOULD be able to refuse to install any elements on their terminal.</p> <p><i>Context Usage and Feed requirements</i></p> <p>R67. The platform MUST provide a set of Context Usage and Feed (CUF) clients to retrieve context information from clients dynamically. Such clients will collect all the available row data from the device.</p> <p>R68. One CUF client MUST be installed on the user device to provide context feed functionalities.</p> <p>R69. The CUF client MUST provide an interface to help the user to configure their credentials.</p> <p>R70. The CUF client MUST provide an interface to help the user to allow/deny the collection of any information about them.</p> <p>R71. The CUF client MUST envelop the data in a XML document.</p> <p>R72. The CUF client MUST have access to the OPUCE platform to send the collected data.</p>

	<p>R73. The CUF client MUST send the document with the context information to the platform to be stored.</p> <p>R75. The user MAY stop to send their own data to the Platform in any moment.</p>
User	<p><i>Remote Service Management</i></p> <p>R57. The user MUST be registered on the Platform and MUST have a SIP account.</p>

Table 4. Requirements classification by platform modules and end user device

6. Conclusions

This document justifies one of OPUCE project main objectives, which is the provision of a infrastructure where end-users and third party service providers and developers can build innovative and integrated services in an easy and interactive way.

The present document mainly accomplishes the tasks of identifying scenarios and ambiances, including on-demand adaptive services and end-users creations, as well as defining a series of requirements that will guide the technical modelling of OPUCE components and platform.

Several scenarios and use cases have been introduced where users are able to create services and deploy them in heterogeneous environments and ambiances, allowing these services to be accessed in a seamless way by a multitude of devices connected via different networks. This will have tangible benefits to end-users as it will enrich services variety and relevance to their needs, and to service providers as it will ease the service creation and extend the scope of services that can be offered. In such environment, the necessity of a common platform becomes the basis, since without it, each organization is left to build their own set of proprietary business protocols, methods and services, increasing costs, times and efforts and leaving little flexibility for true services collaboration.

Finally, the collection of requirements identified from the use cases has highlighted the relevance the OPUCE innovation challenges, and confirms the fact that services described in the deliverable are envisioned as short-lived services, that are adaptable to the environment they are deployed in. This way, they will be greatly benefited by the OPUCE platform.

Annex A. Examples of OPUCE services

A.1. Auto-conference service

Autoconference is a service in which the platform initiates a call or video-call for a group of participants, the conference starts once all participants reach a determined status after a specified time.

The user of the Autoconference service must configure a couple of parameters at configuration time (before execution):

- The user must introduce the timeslot in which the service will be aware of participants's presence status in order to start and launch the conference.
- The user must introduce the list of participants in the conference and the presence status they must reach in order to start the conference. The end-user of the conference service introduces the participants' data (their login name in the OPUCE platform or the SIP URI if one participant isn't registered in the OPUCE platform).

In this way, the conference starts once all participants reach a determined presence status after the specified time. The system will initiate the session with all the participants involved using the appropriated capabilities for each of these users (presence status, terminal registered, video, etc.).

The auto conference service will be considered finished once the conference taking place is finished. However, it could be reused to organize future auto conferences.

The base services that must be available in order to compose the Autoconference service are the following:

- *Presence module*: This base service is used to obtain information on the Presence state of the participants in the conference.
- *Calendar module*: This base service is used to select the date and time-slot for the conference to take place.
- *Contact list module*: It allows the user to introduce the list of participants in the conference.
- *Video/audio call*: It enables the establishment of the video conference.
- *Messaging module (optional)*: This base service is used when the organizer of the conference selects the option to notify the participants of the conference that will take place.

OPUCE features present in this service:

- *Personalization*: At configuration time (before execution) the user is able to select: the desired timeslot for the conference to take place, list of participants, presence state to start the conference
- *Context-awareness*. The user context is taken into account at the moment of call establishment. The participants in the auto conference should authenticate themselves in the terminal they are using at the moment the video call will take place in order for the service to assign the available capabilities to each participant. E.g. If a terminal does not have video capabilities, the service won't assign video to that specific participant.

A.2. Advanced Reachability service

Mrs. Smith is a head manager with a very busy agenda. Sometimes she gets a hard time attending all of his phone calls and sometimes she misses some important calls because of her overloaded agenda. Therefore, her secretary decides to solve this mess by creating a service with the OPUCE platform.

Since the secretary is registered in the OPUCE platform, he decides to set up a service that will efficiently manage all incoming phone calls depending on where Mrs. Smith is and on who is calling. When receiving a phone call, the service will take into account his Presence status, the type of contact calling so the system can react accordingly, and other preferences that Mrs. Smith should complete, such as diverting the call to her secretary under certain circumstances.

Considering all these parameters the service will let the call reach Mrs. Smith, divert the call to her secretary, let the voice mail answer, reject the call, etc.

The Advanced Reachability Service uses the following base services:

- *Presence*: The end user can modify his/her presence status to modify the behaviour of the service. For example, if a meeting has ended before the expected time, the user can modify his/her presence status to point out his/her new availability.
- *Phone calls manager*: It receives the incoming calls and, based on the information provided by the other modules, takes the consequent action.
- *Contact list*: It contains the contact list information of the end user, so the service can difference between relatives, co-workers, etc.

OPUCE features which are present in this service are:

- *Context-awareness*: The user context is taken into account at the moment of call establishment. Personalized automatic management of incoming calls based on presence status and activity information.

A.3. Zaragoza International Exposition 2008

Mr. Mancilla is working for the Organization of the Zaragoza's International Exposition 2008, and his boss has entrusted him with the creation of an innovative service to offer to the people who will attend the Exposition. Using this service the end user will be able to interact with a map of the Exposition watching all the information about the buildings that compose it, the events schedule of each of those buildings, and the activities that are taking place in that moment, receiving notifications about current events. At the same time the user will have access to a hotel location service and to the yellow-pages of Zaragoza.

The Zaragoza Exposition service uses the following base services:

- *Location service*: This block returns the position of the user so the service can place him/her within the map of the Exposition.
- *Map service*: This module is configured to show the map of the Zaragoza Exposition.
- *Agenda service*: It is configured and loaded with the corresponding information about the exposition events and activities. Combined with the Map Service, it provides information on where these activities are taking place.

- *Yellow-pages service*: It is configured to provide the Yellow Pages services of Zaragoza.

OPUCE features/ use cases present in this service:

- *Sharing*: The creator of the service (Mr. Mancilla) shares the new service he created with the visitors coming to the Expo.
- *Recommending*: The users who access to the Expo premises are recommended to use the service by Mr. Mancilla.

A.4. Meeting Agreement

End-user Jon wants to arrange a dinner with a group of friends. It is not usually easy because the group has more than a dozen people, raises a lot of incompatibilities. This is usually a long process requiring many emails, phone calls or MSN talks to achieve a final agreement on date, time and place.

As added restrictions, there is one friend which uses a wheel chair and several that are heavy smokers, which is critical when deciding which restaurant to choose.

So this time Jon will compose a service on the OPUCE platform to manage all the process. The service is based on polls to all the friends to select the restaurant.

The Meeting Agreement service uses the following base services:

- *Public Place Searcher service*: It is configured to search for a restaurant that fulfils the conditions introduced by the users attending the arranged dinner.
- *Poll service*: It is configured to collect the preferences of the users about the type of restaurant and its location. The result of this Poll service is used as an input parameter to the previous Public Place Searcher service.
- *Contact List*: It contains the list of friends that will go to the restaurant and will therefore answer to the poll.

A.5. Massive On-line Multiplayer Game Clan Community

Alice is a young smart girl that likes to game on line with massive multiplayer games. She wants to create and share personalized services focused on the needs of her multiplayer game clan. In particular, she has in mind a service to ease occasional interaction of clan members in the real world, based on localization data. She wants a service to discover if two or more members of the clan are closer to each other than a distance defined by the leader, for example 5km; in that case, if both of them are not in the blacklist of the other user, they are notified about the possibility of being in good company.

The message will be sent in a suitable format for the device that the member is using. The possible communication ways are mail, MMS, SMS or any other resource depending on the devices associated to these clan members. No anonymous messages/calls should be allowed, so the leader uses her personal mobile number or email as notification issuer.

Alice, the clan leader, creates and administrates an OPUCE group of users containing the clan members which requested the service activation. The service polls the members position each time interval selected by Alice (an hour, for example), then notifies the members close to each other of their proximity, according to the type of terminal used.

Users can also dynamically set a status about their visibility, if members set the status to "Invisible", they cannot be tracked or receive notifications.

The Massive On-line Multiplayer Game Clan Community service uses the following base services:

- *Public Place Searcher service*: In case some members are found in the proximity, this module provides them with a suitable public place where to meet that is also close to where they are.
- *Poll service*: It polls the members position each time interval and checks if they are close enough to notify them of each other's presence.
- *Contact list*: It is used so the creator of the service can determine which of his/her contacts will be invited to use the service and also notified about the other's presence.

OPUCE features which are present in this service are:

- *Personalization*: At configuration time (before execution) the user is able to select: issuer address, list of participants, members distance.
- *Context Awareness*. The user context is taken by the position and device type when the service sends the notification.

A.6. Recipe on-line ordering

Bob is a fan of kitchen recipes. Once logged into the OPUCE SCE from its broadband connection at home, he finds that a friend of him donated something that can be useful: a brand new component (developed by his friend) that offers an interface to Google Base to provide recipes that match some tags. He immediately sees how to use it: he easily combines this component with an SMS notification service and his favorite online shop service to buy the required ingredients and make them delivered to his home.

He's now satisfied with her service that will save him a lot of time...and memory for remembering all the steps! He saves the service on the OPUCE service portal and then deploys his service on the OPUCE platform so he can use it immediately, and every time he wants.

Using the service he created, he is notified anywhere by SMS whenever a new recipe related to "tuna" (he's a fan!) is put on the base, and if he likes it (by answering to the SMS), then the service directly orders the required ingredients online to be provided home.

Bob is very proud of his service so he decides to share it with other people that might have the same interests. First Bob makes the service available to a list of his friends and then he decides to make the service public.

The OPUCE Platform notifies users possibly interested in the service (based on profile, similar preferences or interest in some metadata) and makes it available for them on the portal so they know they can execute it as well.

Bob also assigns some metadata (tags) to the service so that the service will be indexed dynamically through multiple facets, and targets a lot of communities. Moreover, Bob allows his friends to edit and improve his service facing their needs (maybe someone would be willing to receive emails instead SMS, for instance).

The Recipe on-line ordering service uses the following base services:

- *FeedReader module*: it notifies the user of any new recipe related to his/her previously introduced preferences.
- *SendSMS module*: used at service execution to send SMS.
- *ReceiveSMS module*: used at service execution to receive SMS.
- *e-Commerce module*: it is used to contact the shop online where to order the necessary ingredients.

OPUCE features which are present in this service are:

- *Sharing*: The user shares the service he creates with his friends and family
- *Recommending*: The OPUCE Platform notifies users interested in preferences similar to Bob's
- *Personalization*: At configuration time (before execution) Bob is able to select the list of relatives and friends, mobile number, preferred online shop.

A.7. Location Based Photo Sharing Album – Mobisaic

Main Scenario: Creation

Peter loves travelling and photography. He loves discovering new places and capturing special moments and places. The day before his journey to China he has a new idea for storing his photos. Since he has just bought a digital camera with Wifi connection support (or a mobile phone with a good camera integrated), he would like to upload their photos in the moment he shots them, to a repository, but associating the image to his location. The storage service allow multiple kind of photograph files, associate a location information to each file and add marks describing the place: city, street, monument, beach, etc.

Once back at home, or from the hotel, Peter considers to watch those photos already stored when travelling, using his laptop accessing a map or navigating through a satellite view of the world, like Google maps service. He can see satellite images of the cities (even the street where the photos were taken) or places that he has visited. When he makes zoom of zones he has visited, a mini image and marks (city, street, monument, beach etc.) of the photos he made in that place appear. Then, selecting that mini image, he can view a bigger image or download them to his computer.

Alternative Scenario: Upgrade

(Note: this alternative scenario takes place once the main scenario has been deployed and made available to users)

Later, Thomas edits the previous service to enhance it. Thomas is an amateur photographer, but he also loves and knows about new technologies. He always looks for services related with photography, and he finds a service created by someone called Peter. He subscribes to it and tries it.

Thomas considers the service very useful and attractive, but he considers that can be upgraded. He would like to invite groups of friends to access to collections of photos uploaded during a journey or to notify them when they are in a place where one of his friends made and uploaded a photo. He remembers about when his friends and he visited Germany.

Thomas would have liked not only to upload to Peter's location-based photo album service his photos, but also to invite those friends who went with him to see them. He would also notify (via SMS, MMS in mobile devices or whatever) a Thomas' group of people (for instance, his parents, friends or colleagues) about that album when they visit the same city or place where Thomas made those photos.

Thomas notices that the service can go a step further. He considers that the information available in a certain place should be images but also text, videos or sounds. And that when he uploads some information when he is in a certain place, it could be available to anyone that wishes to know what people has uploaded in that place (for example, comments about one restaurant in that place or history information about a monument).

The Location based Photo Sharing Album service uses the following base services:

- *Maps service*: This base service provides a set of satellite images. It is used to mark those places where the user has uploaded some kind of data.
- *Location service*: This base service is used to retrieve the geographical position of a user.
- *Contact List module*: It is used to create and manage contact list for friends, family, colleagues, etc. It also allows generating an invitation for a certain group to access to some kind of information.
- *Photo storage module*: It used to store the pictures and the information related to them, such as date, place and other relevant information. This service is used to store data (images, text, video etc.). It can be a simple database, a distributed database, etc.
- *SendSMS/MMS service*: This service is used to send SMS or MMS to mobile devices. It will act as an interface with the SMSC or MMSC of the mobile network of the operator.

OPUCE features which are present in this service are:

- *Sharing*: Peter shares his services with the other OPUCE users. It is necessary since Thomas will improve it later.
- *Recommending*: Peter recommends the service to those users interested in photography.
- *Personalization*: Only in the service upgrade, Thomas makes a contact list of the people who will be informed when they are near to a place where Thomas took a photograph.
- *Context Awareness*: The service takes the context of the photo device (camera with Wifi, mobile...) and the position where is taken.

A.8. The Big Thinker

I (TBT – The Big Thinker) need my own service to help me manage my small consulting company (“the big think”) that provide and develop new business ideas to other companies. On the other hand the service should improve my quality of life by increasing (and improving) my leisure time with my family and friends while it increases my productivity.

My company is a fast growing company in a new emerging market that makes me travel a lot. Even if I'm away from home I want to remotely join my family for meals and leisure activities like online gaming. The service facilitates and makes it easier the way I participate in an activity I enjoy, by using my personal profile data and that of my family, as well as context, schedule, mood and ambiance.

At the end the service will make my life easier and my time better spent, as well as that of my company collaborators and family members, since, everyone has the same objectives, and thus we don't lose time with unnecessary mistakes or misunderstandings about what we want to do.

The Big Thinker service uses the following base services:

- *Calendar*
- *Content Management*
- *AAA*
- *Messaging*

OPUCE features which are present in this service are:

- *Personalization*: Adaptation of a service to the user preferences.
- *Context Awareness*: Adaptation of a service to the environment (multi modality)
- *Service sharing*: Publication of a service

The TBT service is composed of different sequential sub-services that include several use cases:

- The Service Creation use case including Features procurement, service composition, service publication and subscription by customers and friends.
- Idea Profiling and Idea Board Management Session Arrangement Use Case (Focused on Profiling and Matching. context awareness arrangement).
- Idea Board Management Session (focused on context awareness communication).
- Customer/Idea Procurement and Match. (focused on Context Awareness Info/Join Me).
- Meal and Gaming with Remote Family and Friends (focused on Context awareness online leisure).

SERVICE CREATION SCENARIO

Ambiance and Usage Roles

In a first stage I have to define the main ambiances and usage roles involved in the service.

The service will be used in two different Contexts /Ambiances: Work and Family. Each Ambiance will be identified by different variables, including:

1. usage places
2. usage time
3. usage activity
4. usage roles in the service relationships:

- a. *Think Manager role (myself)*: help me running the company providing me tools to manage my group of thinkers and sell ideas to other companies
- b. *Thinker role (my employees)*: help each Thinker manage the full life-cycle of each idea
- c. *External Expert (outsourcing)*: external expert that provide support in evaluating and selling business ideas
- d. *Think Customer role*: the Think customer subscribes the Service I create to manage idea requests, support management, etc
- e. *(my) Husband role*: help me keep a decent quality of life and avoid me being addicted to an workaholic and sad lifestyle, by giving more attention to my wife, remembering me about important events, providing me tips for good gifts to her, managing my communication with her, managing our own event arrangements, notification about her relevant context status, ..
- f. *(my) Parent role*: giving more attention to my children, remembering me about important events, providing me tips for good gifts to them, managing my communication with them, managing our own arrangements, notification about their relevant context status, remotely play with them.
- g. *(my) Children Role*: manage their communication to me or my wife, managing their request for family arrangements, having parent support according to his context status, remotely play with me or my wife.
- h. *(my) Wife role*: manage her communication to me or our children, managing their request for family arrangements, having my support according to her context status, remotely play with them
- i. *Leisure Activity Provider* (e.g., Restaurants, online Gaming, etc.)

Features Procurement

Then I write down a set of functionalities needed for each role and then I try to find them in the Opuce Feature repository. For Features not currently available in the repository I ask the system to look for them in other platforms by filling in a template describing the feature profile I'm looking for. I'll be notified about features matching my request. I also can define my own Feature by the composition of existing features. Initial list of features are provided below (to be completed with use cases).

Work Context

1. Content (idea) Management:
 - a. Content (idea) life-cycle management
 - b. Content (idea) directory
 - c. Content (idea) profile
 - d. context (more dynamic info from ideas like its stage, inputs from thinkers)
2. Work Calendar Management (Context Awareness Arrangement)
 - a. Company calendar
 - b. Idea calendar / Idea implementation calendar Management
3. Ideas Match:
 - a. Match requested ideas with existing ideas

- b. Match new created ideas with customer profiles (Business needs)
- 4. Online internal collaborative sessions to (Context Awareness Communication):
 - a. evaluate new ideas or ideas request
 - b. work on ideas on sale
 - c. work on ideas under implementation
- 5. Online collaborative sessions with clients to (Context Awareness Communication):
 - a. sell ideas to clients
 - b. work on sold ideas with clients
 - c. support ideas realization

Family Context

- 6. Family Context Management (Context Awareness Arrangement) including:
 - a. I want to manage the time I spend with my family and be notified when I don't spend enough time ...
- 7. Family Calendar Management (Context Awareness Arrangement)
- 8. Family (and friends) online session (Context Awareness Online Leisure)
 - a. Remote meal sharing
 - b. Remote online gaming

Crossing Contexts

- 9. avoid conflicts among
 - a. Working Calendar and Family Calendar (Context Awareness Arrangement)
 - b. Working Communication and Family communication (Context Awareness Communication)
- 10. based on having different contexts for each role:
 - a. if the context is mainly business I'll give more priority to business issues
 - b. otherwise I'll give more priority to family issues
- 11. The way how conflicts are managed should also take into account the amount of time I spend in each role

Service Composition

Each feature is defined by a set of rules that define the behavior of the feature in terms of conditions and actions. This is provided by a nice graphical tool where each feature is a graphical object that I can link to other objects and thus build a nice workflow. Then I can zoom in each feature and detail its behavior in terms of a workflow of actions ruled by events - conditions.

For example, regarding the Context Awareness Communication Feature, for each Company role and family role I define a set of rules to manage incoming and outgoing communication sessions including:

- 1. Communication with clients
- 2. Communication with family
- 3. Communication with/among thinkers

4. Incoming Notification for Ideas matching
5. Incoming Notification for Company Calendar Alerts
6. Incoming Notification for Idea Project Implementation Calendar Alerts
7. Incoming Notification for Family Calendar Alerts
8. Incoming Notification for Idea

I also define a set of workflows among features for the different types of sessions. The workflow is managed by graphical commands during ongoing sessions.

Service Publication and Subscription

The service is published and advertised among customers and free lancer experts after the composed service is approved (by the Service Auditor).

PROFILING AND BOARD MANAGEMENT IDEA SESSION ARRANGEMENT SCENARIO (FOCUSED ON PROFILING AND MATCHING, AND CONTEXT AWARENESS ARRANGEMENT)

I'm travelling abroad for 1 week meeting several customers. One Business Idea is conceived by one of my collaborators. The Idea is automatically profiled by the service in terms of feasibility and business value (content profiling and matching to business profiles from customers). Since the idea profiling got a good Business Value classification, an Idea Board Management Session is automatically arranged for final evaluation and approval. For this arrangement to take place a previous list of attendees including demanded profiles for external experts and information material is recommended by the service and approved by myself. The service searches for the right experts matching the requested profiles. After agreements are achieved with experts found, the session is scheduled according to context availability from all invited participants (including demanded terminal capabilities) as well as requested material availability.

Sequence of Actions

1. TBT asks his secretary to arrange a business trip for 1 week to visit European customers.
2. TBT secretary collects all information about the trip (transport means, departures/arrivals time, locations, hotels address, etc) and inserts it in TBT Calendar for approval.
3. TBT checks the trip arrangement and approves it.
4. TBT is reminded in advance several times about the departure time. These calendar reminders are dispatched to the most appropriate connected device according to TBT Context and they are triggered according to different collected information (e.g., traffic, distance to airport, etc) used to estimate time to reach the airport and get in the plane. Reminder Examples are:
 - a. Wake-up alarm. It keeps triggering the alarm until the TBT Context collected information indicates that he is in the shower
 - b. Update reminders: while TBT is at home it gives more information about the traffic, time estimated to reach the airport by taxi, weather forecast for the different places TBT is travelling to (he may choose the most appropriate clothes), etc
5. TBT gets into Taxi and receives more information about the trip, calls to the check-in airport service which performs the check-in, including luggage check-in (RFID). An electronic boarding card is issued and sent to his mobile device.

6. TBT Context is updated and the service uses the boarding card Id to subscribe for trip information (airport land services and airline operator). Notifications received from these subscriptions are used to keep TBT Context updated and to warn TBT about any relevant trip information.
7. TBT arrives to the airport, leaves the luggage that is immediately identified by the electronic boarding card (eBC) associated RFID, passes through the security staff using the fast lane (recorded video and eBC gives makes him more trustful for security authorities) and directly goes to the boarding gate. The eBC is checked and TBT gets in the plane.
8. In the meanwhile TBT context is updated according to information received from airport land services: the user will be unavailable and on flying status till the service receives information that the plane has landed.
9. TBT Context is published to TBT calendar trip event status which is updated according to trip progress and time to land estimation. This way authorised and interested people may get accurate information when consulting TBT calendar.
10. After the plane lands TBT context information is updated, including new location information, availability to receive communications, time zone, etc
11. The context is published and notified to his closer buddies namely family members will know that TBT had a safe trip and what is his time zone.
12. TBT calendar trip activity status is updated to new status. Next event in the calendar that had a dependency on the trip event end is triggered: call to family members
13. The call is set up and TBT Context is again updated: now he is on the phone with his wife and he won't be available for other activities during the call.
14. This context activity is published to TBT calendar with an estimation of the call duration: 5 min.
15. The Call finishes: TBT Context and Calendar are updated
16. On TBT 2nd day trip, one Thinker publishes a new idea: new playground design (types of ideas are configured by TBT along with all needed information that contributor has to put):
 - a. Taking into account Thinker device capabilities, the service generates a form for thinker to insert idea profile including: contributor id, location, type, field (in terms of field of interest to match profiles) etc.)
 - b. Based on initial inserted info, Service generates more detailed forms – business cases references, keywords/tags, exact location, evaluation/development plan (tasks organization, resources, duration, profiles needed,..), expected costs, expected revenue, needed raw materials, etc
17. Service creates internal representation of idea (idea profile) based on collected data from Thinker including Idea Agenda with Meta profiles needed for each planned task in the Idea Agenda: internal Thinkers, external expertises (e.g., needed Plastic manufacturers), customers, etc.
18. Idea Agenda first task is Idea initial evaluation:
 - a. The Service creates a list of internal Thinkers, external experts and customers for evaluation purposes.
 - b. The service asks internal and external Matching feature to look for user profiles that match the meta profiles by using the internal and external User Profiles repositories (see **Error! Reference source not found.** in previous Use Case for further details). The match takes into account availability status of each user

calendar according to Idea Agenda – for instance very busy users during scheduled Idea Evaluation Task won't be included in result, or will be included with low priority.

- c. Matched profiles are recommended to Thinker that selects some of the recommended profiles and asks the service to invite them for evaluation
- d. Service collects data evaluation from Idea Evaluators that accepted evaluation invitation (see **Error! Reference source not found.** in previous Use Case for further details)
- e. Based on data collected from Thinker and evaluators, Service performs initial evaluation (depending on type of idea) in terms of needed/already available resources, costs/revenue, needed work/man power, etc
- f. Evaluation process takes place now, which produces some sort of report and final grade

19. The second Task in Idea Calendar is initial Idea approval by TBT:

- a. Service notifies TBT that new idea has been created; it also provides TBT with short description and short report along with its business grade.
- b. TBT accesses Idea Profile management and uses detailed view to analyse it.
- c. TBT calls Idea Publisher and clarifies some issues that are recorded and attached to Idea Profile
- d. TBT accepts the published idea with a good rate
- e. Idea Calendar and Context is updated and Publisher (Thinker) is notified about it.

20. Since the published idea is approved for discussion by TBT the next task scheduled in Idea Calendar is Idea Board Management Session arrangement task:

- a. Service accesses profile/context/calendar information of User Profiles allocated to the planned Board Management Task and does cross check on availability taking into account the deadline to close the task. It returns a list of dates that are inserted in Idea Calendar as candidate dates for the session. Each date is rated taking into account the ones that best fit into attendees calendar
- b. Service asks (invites/notifies) TBT to choose a day
- c. Service performs normal Calendar event invitation/negotiation (negotiation seems possible, but probably unwanted) with users,
- d. Idea Calendar and Attendees Calendars are updated
- e. Resources needed for Idea Board Management (session conference bridge, content, etc) are reserved according to scheduled session in Idea Calendar and foreseen terminal capabilities of session attendees

IDEA BOARD MANAGEMENT SESSION (FOCUSED ON CONTEXT AWARENESS COMMUNICATION)

The Idea Board Management Session is set up according to the previous arrangement. The idea creator presents the idea by using different material including slides, video clips with business cases and forecasts supporting the Idea business value.

Customer profiles that matched the idea i.e., those that are potentially interested, are presented and edit by all participants. The business value is calculated for each matched customer.

Parallel sessions are set-up with external experts and potential customers according to their context availability. Participants browse and are moved among sessions. External experts

give their opinion mainly on idea implementation feasibility in terms of costs and time supported on multimedia material shared among all participants.

During the session I receive several requests to set up a communication session. The service automatically handles these communications according to the rules I set up for this context: some are reject, others are forwarded to other colleagues of mine, etc. I accept one: my son calls me from the college. He was crying because he got hurt when playing with other kid. I speak to him to calm down him and called my wife to help me. I had to move to the work session but I kept monitoring the session status and specially my son status.

Finally the idea profile is completed, approved and published.

Sequence of Actions

1. Service reminds each session participant about scheduled session according to session reminder settings, e.g., 15 mins before it starts
 - a. Idea Calendar inner trigger indicates that reminder should be sent
 - b. For each participant looks up for the most appropriate way to notify participants about the scheduled session taking into account user and registered terminal context/profile/presence information
 - c. Reminder is sent
2. Idea Calendar triggers the service to initiate session set-up. The service invites all participants to connect to the Session bridge taking into account each participant context, profile and available terminals:
 - a. according to terminal capabilities the service creates different types of Communication Handlers (for instance user that currently has only simple cell phone doesn't have interest in video data if available)
 - b. for each type of users a separate room is created: Main Management Board session and Experts Session
 - c. users are invited to join conference sessions
 - d. TBT is the moderator of all sessions and is the first one to be invited by the service, and he welcomes each participant joining the session
3. TBT Context is updated and published in his calendar as busy in a meeting planned to take 1 hour long
4. Session starts and Idea is discussed
 - a. TBT introduces the session subject and the planned agenda. He asks his secretary to take notes for the minutes.
 - b. The secretary starts recording the session introducing indexed notes along the recorded session according to the presented agenda.
 - c. TBT gives the floor to the Idea Publisher to shortly describe it
 - d. Idea Thinker publisher selects Idea Data Material from the Idea Directory and starts presenting the Idea along with Slides, videoclips and recorded audio material
 - e. All participants are muted but TBT. When someone speaks, the icon representing participant voice activity blinks but his voice is not heard. The Idea Publisher (the current session moderator) may un-mute to let everyone listen.
 - f. Idea profile is discussed and jointly edited by the group (a la Google Docs)
5. TBT browses between board room and experts room to discuss some details
 - a. One technical issue is raised that demands the assistance by an external expert. The external expert profile is selected from the list of experts associated to the

- idea profile and his context is checked: he is available to join the session. TBT asks the Expert to join a parallel session along with two thinkers of the main board management session. The remaining main board members keep discussing and completing the Idea Profile.
- b. While TBT discusses the technical issue in the parallel expert session, he is monitoring (listens, watch, read to) what is happening in the Board session and he punctually gets involved in it
 - c. During the talks with experts he decides to change expected cost of two tasks in the Idea Profile
 - d. The idea evaluation is updated and the Idea view (for users that has this screen open) is refreshed
6. TBT switches back to board room. After a while three calls invitations are addressed to TBT. The way how the call is handled takes into account the current TBT Context.
- a. 1st call : from = Thinker – subject = „ social event for new employees”, type business
 1. service calls some evaluation component which tests call information against some predefined values, for instance call subject(? If possible), originator, priority
 2. evaluation component asks context/profile sharing component about specific information/settings
 3. asks calendar if TBT has scheduled some time(later) on that subject(or very similar)
 4. evaluates given date
 5. after evaluation decision is made, this call has low priority, its not urgent, those matters can be attended by TBTs secretary – call is forwarded to her
 6. service looks up secretary status/presence and automatically dispatches this call to her.
 - b. 2nd call – mother in law – subject: „dinner”, priority high, type: family event/dinner. same steps are performed as above but:
 1. call has high priority(+)
 2. TBT is family man-he values time spent with family(+)
 3. calendar indicates that he did not spend much time with his mother in law (++++)
 4. TBT dislikes his mother in law, she puts her nose in matters she should not (-----)
 5. Action: automatically forward call to TBT voice mail
 - c. 3rd call TBTs son is calling: subject: “school”, type: emergency call (?). Same steps as above but:
 1. son is calling(+)
 2. subject is school – one in which TBT is interested in(+)
 3. type is emergency(highest priority/grade(?))
 4. Action: asks TBT for permission and put call on hold
7. Service looks up means of contacting TBT (context/profile presence(?)) – voice is available (and preferred since its a call)
- a. Service answers the call, plays the current busy status and asks TBT son to record a small description of what is happening
 - b. Service dispatches a link to TBT Communication Handler with the recorded son’s explanation and (textually) asks him to receive this call
 - c. TBT listens to his son explanation in a whispering mode (the other participants don’t noticed it).
 - d. TBT decides to accept the call while keep monitoring on going working sessions

8. TBT son's call is finished and he TBT's audio communication is resumed
9. At some point TBT approves idea. service generates idea report that is attached to the session minutes. The recorded indexed session minutes is attached to the Idea Agenda and the link distributed to participants. This way recorded session minutes may be consulted and searched for specific discussed topics by using audio search engines (e.g., <http://www.annodex.net/>).
10. service terminates sessions and board session is closed. TBT context is updated and published to his calendar.
11. Idea is publicly published into public Idea repository. Idea profile is matched with customers profiles
12. Matched customers profiles are notified about the new Idea taking into account its context e.g., customers are in a good working mood

CUSTOMER/IDEA PROCUREMENT AND MATCH (FOCUSED ON CONTEXT AWARENESS INFO/JOIN ME)

By chance one Local Customer got interested with the just published Idea. They urgently want to get an agreement and an adhoc physical meeting is arranged with remote co-workers and experts according to their context availability and location.

During the session I'm interrupted several times by my wife to arrange the remote shared meal: I look for nearby remote meal enabled restaurants, that match my foods tastes and arrange it according to the context of my remote family (synchronise local meal arrangement with my meeting and my meal in the restaurant). I got a look on the samples plates that matched my profile. It looks nice but I had no idea what it was (exotic Brazilian plate).

The status of the meeting (and myself) and also the time to arrive to restaurant is simultaneously monitored by my wife and the restaurant manager in order to get both ready at the same time according to the meeting.

The time is running out and the customer is hard to convince. I realised I should start thinking how to get to the restaurant as fast as possible. While the costumer is arguing with the external expert I asked the service for a new feature - taxi arrangement – where I introduce the address of the place where I am the restaurant address and my context feed with appropriate filters to provide the needed information about my context to the taxi service.

The idea is finally purchased by the customer and the contract signed. I get out the building and the taxi is there waiting. I move to the restaurant.

Sequence of Actions

1. Local customer gets interested by an idea and arranges a physical meeting with remote co-workers and experts. Service:
 - a. Looks up through contexts/profiles to choose the participants of the meeting
 - b. Search the best located ones for the meeting (location)
 - c. Search the available ones (availability)
2. During the meeting The Big Thinker (TBT) is interrupted several times by his wife. Service:
 - a. Evaluates the call in terms of subject, priority and origin (accepts it because it is from his wife)
 - b. Established session with the wife during the meeting in the suitable media
 - c. Changes status/availability/context information of participants

3. BT looks for the meal arrangement according to his context, his family context and his food tastes. Service:
 - a. Search remote meal enabled restaurants (InfoMe about desired subject)
 - b. Match available restaurants according to TBT context and his remote family context (location of every member involved, availability of the restaurant, price, if it is a “family” restaurant or a “romantic” one, etc).
 - c. Match available restaurants according to TBT food tastes
 - d. New participant is added, the Restaurant Manager in an alone session together with TBT.
 - e. Media Data Transfers between TBT and the restaurant manager to watch food samples that match TBT tastes and to have a look around the restaurant environment. If no tables are available to book, another matched restaurant is selected and the same procedure is done.
 - f. As an agreement about the meal is reached, the session is frozen and TBT Family Calendar is updated with the meal arrangement
4. The BT meeting status and the time to arrive to the restaurant is monitored by both the wife and the restaurant manager. Service:
5. TBT Context is published to TBT calendar trip event status which is updated according to trip progress and time to land estimation. This way authorised and interested people may get accurate information when consulting TBT calendar.
 - a. TBT Calendar is updated with estimation of meeting duration according to meeting progress published by TBT
 - b. This way authorised and interested people that subscribed for TBT context will be notified with TBT Calendar updates (sending of the context)
6. As time is running out, BT asks for a taxi arrangement. Service:
 - a. Ask a taxi locator in the session, as an external service
 - b. The external service is added to a parallel session
 - c. Address, Context, Time to live the meeting and Location is sent to the Taxi Locator
 - d. Match suitable taxi according to this criterion
 - e. Chosen Taxi is inform
 - f. When the taxi reaches the place, the taxi locator gets out of the session
7. Idea is finally purchased and contract signed. Service:
 - a. Updates idea information/profile
 - b. Customer is added to Client list
 - c. BT ends session
 - d. BT context is changed to “moving” as he is in the taxi, as well as his location.

MEAL AND GAMING WITH REMOTE FAMILY AND FRIENDS (FOCUSED ON CONTEXT AWARENESS ONLINE LEISURE)

During the taxi trip I check with my family and the restaurant the plan for the remote meal sharing. Everything seems ok. I still had time to watch the meal getting prepared by the cooker and ask him to prepare the meal the way I like.

While talking to my wife she asked me to plan online gaming with friends and the children after dinner. I arrive to the restaurant just in time to remotely join my family.

During the meal I’ve noticed the customer was trying to contact me but my context is clear: I’m with my family, my wife is already not very happy with the time I spend far from the family

and the subject of the session is not urgent. The service automatically handles the call for me and schedules a call back set – up in the very morning.

After the meal the service schedules the online gaming and selects the right gaming friend partners beside my family. I move to my Hotel and I play in my gaming enabled room (the TV set is fantastic). At the end of the game I select a more relaxing music and I start getting sleepy, therefore I ask them they get another partner and I go to sleep . It has been a long day. My family and friends go on gaming without me.

Sequence of Actions

1. The Big Thinker checks the Remote Meal Plan while moving on the taxi. Service:
 - a. Check in one session the taxi trip and the expected time to reach the restaurant.
 - b. Check in another session the time the meal is going to be prepared.
2. He watches the meal getting prepared and asks the cook to prepare the meal the way he likes. Service:
 - a. Close the taxi session as everything is OK.
 - b. In the restaurant session, invites the cook.
 - c. Start a video and audio communication to see how the meal is getting prepared and to talk with the cook.
 - d. Session is closed or maybe frozen (depending on if it is going to talk with the cook later or not).
3. After that, he talks with his wife and they decide to plan an online gaming session with their children and friends. Service:
 - a. Start a session with the wife.
 - b. The service plans a gaming session right after the dinner in TBT Calendar
 - c. The Big Thinker adds his children as participants of the gaming session.
4. During the meal, the customer tries to contact the big thinker, but as the context is “with family” and the call is not urgent, a call back is scheduled for the following morning. Service:
 - a. The children are added to the session with the wife to start the remote meal.
 - b. The context of every participant is changed into “with family”.
 - c. The service manages all the incoming calls due to the context of The Big Thinker, the person calling and if the call is urgent or not.
 - d. The service decides to schedule a call back for the following morning, as the call is not priority.
 - e. Both the Big Thinker and the customer are informed of the scheduled call back.
5. After the meal the service is triggered by the scheduling gaming session in TBT Calendar and chooses the best gaming partners. Service:
 - a. Look up through context/profile to choose the participants of the gaming session.
 - b. The chosen ones are informed to join the gaming session.
 - c. The ones that decided to join are added to the already open session with the wife and the children.
6. The Big Thinker reaches his hotel and takes part in the gaming session in the hotel's TV set, which is properly prepared. Service:
 - a. As the service detects a suitable device for the gaming session (the TV set) ask The Big Thinker to use this device for the gaming session.
 - b. The Big Thinker agrees with that and the gaming session start.

7. The Big Thinker selects more relaxing music; he feels sleepy and ends the session asking the others to search for another partner. Then he goes to sleep. Service:
 - a. The Big Thinker starts using a music service.
 - b. He decides to leave the session, inform the other partners.
8. Finally closes the session.

References

- [1] OPUCE consortium, *D3.1: Service, service lifecycle and service components specification*, March 2007.
- [2] OPUCE consortium, *D2.3_1: Description of OPUCE platform elements*, March 2007
- [3] OPUCE consortium, *D2.4_1: Description of OPUCE platform interfaces and reference points*, October 2007
- [4] MaCS (Multimedia Communication Services) consortium, *Market Study Report*, 2005
- [5] D. White, JISC funded SPIRE project, *Results of the "Online Tool Use Survey"*, March 2007
- [6] <http://www.flickr.com/>
- [7] http://en.wikipedia.org/wiki/Main_Page
- [8] <http://www.myspace.com/>
- [9] <http://uk.youtube.com/>
- [10] eKiss session: *FORRESTER "User Generated Contents: Telco's role"*, April 2007
- [11] http://www.openhandsetalliance.com/android_overview.html
- [12] <http://www.movilforum.com/servlet/SrvInicio>
- [13] http://www.mono-project.com/Main_Page
- [14] <http://linuxdevices.com/news/NS4147346850.html>
- [15] <http://www.gartner.com/it/page.jsp?id=506328>
- [16] Gartner Dataquest (May 2007)
- [17] A. K. Dey and G. D. Abowd, *Toward a Better Understanding of Context and Context-Awareness*, College of Computing, Georgia Institute of Technology, Atlanta, Georgia GVU Technical Report GIT-GVU- 99-22, 1999.
- [18] G. Kappel, W. Retschitzegger, W. Schwinger, *Modeling Customizable Web Applications - A Requirement's Perspective*, 2000 Kyoto International Conference on Digital Libraries, November 13th -16th, 2000.
- [19] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.