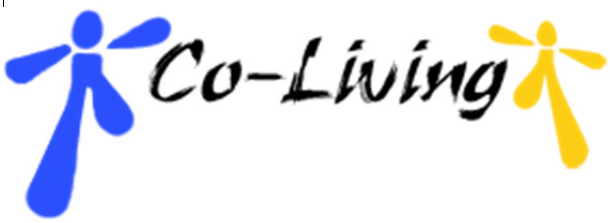

Deliverable reference:	Date:
D 2.1-b	30/09/2013

Title:

Overall system design

Project Title:

Virtual Collaborative Social Living Community for Elderly

Co-Living

Contract no. 60-61700-98-009

Responsible partner:

SINTEF

Editors:

Anders Kofod-Petersen

Approved by:

AAL-2009-2

Classification:

Confidentiality: Project Internal

Dissemination Level: PU

Abstract:

This document is the result of the architecture design phase of the Co-Living system. The design emerges from the underlying mPower platform ensuring a coherent and unified architecture that ensures the required modularity, expandability and interoperability. The guidelines and considerations driving the design process are presented. The overall system architecture and how the system is partitioned into subsystems or components are specified. Each of the major system components is discussed. The document describes how the functionality and responsibilities of the system address the documented user requirements. Finally, the principles and guidelines leading to the specification of the user interface component of the system are discussed.

Keywords: Overall design, Service Oriented Architecture, System architecture, User Interface

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Summary

Task 2.1, “Co-Living System Design” is the task that designs the Co-Living systems. The initial user requirements, as well as those emerging throughout the development process are translated into a representation of software components, interfaces, and data necessary for the implementation phase. The outcome of this task is the “Overall system design” document, which presents how the Co-Living system is structured.

Various software quality indicators are being used throughout the design, development and deployment of the system. The responsibilities of Co-Living are partitioned and then assigned to various subsystems. This document serves as the primary reference for the detailed design of the system’s subcomponents during WP2. These components are: the Social Community network (SoCo-net), the ICT-based services, the security and privacy infrastructure and other supportive structures.

Task 2.1 is performed in two stages, producing two sequential versions of this deliverable. The first was a preliminary design where the overall system architecture was defined, identifying each high level subsystem and the roles or responsibilities assigned to it. In the second stage, a more complete component specification is achieved and interfaces are defined, based on the imposed functional requirements and following the software quality indicators of the first version.

1.2 Role of the deliverable

This deliverable defines the overall design of the Co-Living system. The purpose is to present a coherent and unified architecture that ensures the required modularity, expandability and interoperability. The design emerges from the underlying mPower platform. This deliverable describes the system overview including traceable requirements, design considerations, system architecture and the most important components in detail. The details in this version of the deliverable correspond to work carried out in both phase I and phase II of the project.

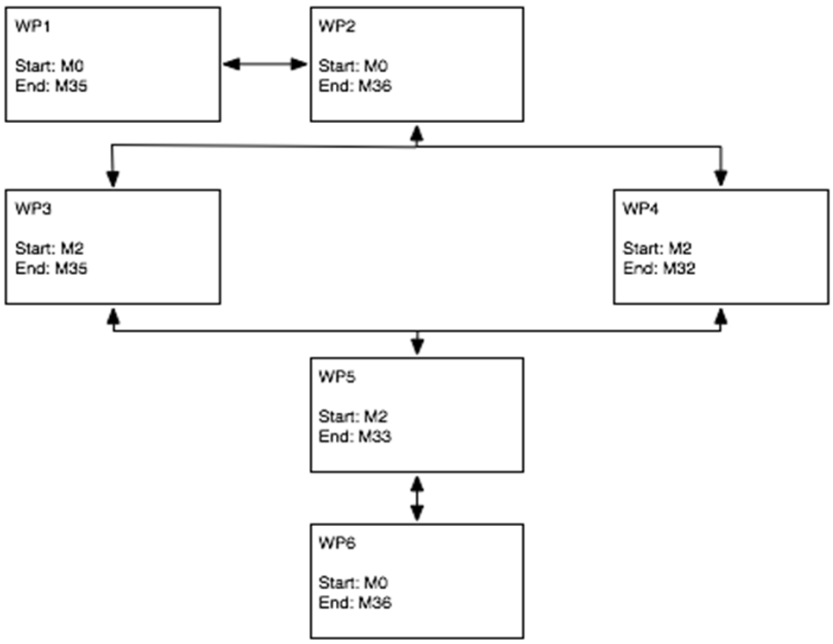

Figure 1: Overall work plan for Co-Living

The overall system design relies primarily on the specifications of use cases and requirements gathered in WP1 (see Figure 1 for an overview of the work package dependencies). The result of the design allows WP3 and WP4 to develop the SoCo-net functionalities and the ICT-based services, respectively.

Phase II of the project is divided into four cycles. Each cycle consists of: the update of requirements (WP1), the design (WP2), the development (WP3 and WP4), the integration (WP5), and trial operations (WP6).

1.3 Relationships with other deliverables and work packages

As described in the preceding section, this deliverable is concerned with the overall system design. Therefore this deliverable acts as the main focus point for the development of the social network functionality, as well as development of the required services populating the Co-Living system.

1.3.1 Work package 1 – End user needs analysis and requirement specification

WP1 main objectives are to analyse and identify the socialisation needs of the elderly; described suitable use case scenarios and define any ethical, privacy and legal considerations for the end user. This deliverable will report on the design following the recommendations and requirements given in WP1.

Deliverable 2.1.b is closely related to the following deliverables:

- **Deliverable 1.1 – Specification of user socialisation needs and analysis and design of the innovative social practice-oriented community model:**

  This initial version of this deliverable describes the user needs in the context of the Co-Living project. The user needs are described in relation to the three main categories of: Care & Wellness, Guidance and Mobility Monitoring. These requirements (see Section 2.2) build the foundation of the design considerations (see Section 3) and the actual design (see Section 4).

- **Deliverable 1.2 – Specification of use case scenarios:**

  Deliverable 1.2 describes the initial use cases for the Co-Living system. Theses use cases form the basis for the system design and future realisations.

- **Deliverable 1.3 – Specification of ethical, privacy and legal considerations:**

  This deliverable details the different ethical, privacy and legal constrains given both on a European level, and for the Netherlands and Norway. Many of these considerations have had a strong influence on the Co-Living design.

1.3.2 Work package 2 – System design and architecture definition

The main objectives of this work package are to construct a suitable overall design that satisfies the requirements in the domain approached by Co-Living The design also includes a design of the SoCo-net to be used, the design of the ICT services exposed and the security and privacy structures.

In addition to residing in the same work package, Deliverable 2.1 is closely related to the following two deliverables:

- **Deliverable 2.2 – Design of SoCo-net and security and privacy infrastructure:**

  This deliverable describes the SoCo-net, and the security and privacy elements. The SoCo-net is the core social network module, whereas the privacy and security elements are enhancements of the mPower security and privacy infrastructure, making it in line with the requirements described in Deliverable 1.3.

- **Deliverable 2.3 – Design of ICT-based services:**

  Deliverable 2.3 contains the design of the ICT-based services within the three categories of Care & Wellness, Guidance and Mobility Monitoring. The document describes the existing mPower services that can be reused, as well as the design of the new services required.

1.3.3 Work package 3 – Social community network (SoCo-net) development

WP3's main objective is to develop and test the SoCo-net module. This work is based on the design effort carried out in WP1 and 2 and described in this deliverable.

1.3.4 Work package 4 – ICT-based service development:

WP4 is charged with the responsibility of developing mobile ICT services. These services are based on the design resulting from the work in WP1 and 2, as described in this deliverable. The ICT services are developed in three categories: Care & Wellness, Guidance and Mobility Monitoring.

1.4 Relationship to other versions of the deliverable

The current and second release of this deliverable establishes the Co-Living system design as it has been developed at the end of phase II. It describes the underlying concepts and considerations that drive the architectural design process and presents at a high level how, based on these principles; the system will meet the documented requirements.

1.5 Structure of this document

The rest of the document is structured as follows:

- Chapter 2 provides a general description of the design process and a summary of requirements related to the functionality and to the overall design of the system.
- Chapter 3 describes many of the issues addressed before attempting to devise a complete architectural design.
- Chapter 4 provides a high level overview of how the functionality and responsibilities of the system are partitioned and then assigned to subsystems or components.
- Chapter 5 presents a discussion of the components described in the previous section.
- Chapter 6 discusses principles and guidelines leading to the specification of the user interface component of the system.

The Glossary and References follow. The Annexes present further reference and background material.

1.6 Deliverable contributors

Table 1: Deliverable contributors

Partner name	Contributor name	Contributor email address
Andago	Idoia Olalde	idoia.olalde@andago.com
Citard	Christophoros Christophorou	christophoros@cs.ucy.ac.cy
Citard	Eleni Christodoulou	eleni_christodoulou@cytanet.com.cy
IPN	Christiana Tsiourti	ctsiourti@ipn.pt
IPN	Paulo Freitas	freitas@ipn.pt
IPN	Jorge Dias	jorge@deec.uc.pt
Philips	Paul Koster	r.p.koster@philips.com
Philips	John Bernsen	john.ac.bernsen@philips.com
SINTEF	Anders Kofod-Petersen	akof@sintef.no
SINTEF	Ståle Walderhaug	stalew@sintef.no
UCY	Dimosthenis Georgiadis	dimos@cs.ucy.ac.cy
UCY	George Samaras	cssamara@cs.ucy.ac.cy

2 System overview

In this chapter we introduce the overall design strategy followed to achieve the goals of the system and present the scope of the solution in terms of the most significant set of requirements that characterize and are addressed by it.

2.1 Design overview

The ultimate aim of the project is the development of an ICT-based Virtual Collaborative Social Living Community (Co-Living) for elderly people, aiming to stimulate and prolong their independent and active living, and social interaction in outdoor environments, contributing thus positively to their well-being.

Co-Living main-sub goals include the development and support of an innovative, social, practice-oriented community model; build around the elderly, and the scaling up of the successfully developed and piloted IST FP6 mPower open source platform to make it applicable to the field of elderly social interaction context. For more details on the objectives of system please refer to the Project Plan (PP) document [1] and the collection of requirements and scenarios collected in the document [3].

The first step in achieving these goals is of course to understand the requirements of the future system users. The design of Co-Living is driven by the requirements of the elderly and also the various care-takers who participate in virtual collaborative network (SoCo-net). In WP1, the needs and requirements of all the stakeholders are well documented in broad terms. These requirements (see Section 2.2 for a summary) build the foundation of the design considerations (see Section 3) and the actual design (see Section 4). However, due to the nature of the solution, user needs and requirements can easily change during the project lifecycle. Thus, it is crucial that the development proceeds as an exploratory process, in close collaboration with the future users.

To address this issue, the project methodology suggests a phased approach for arriving at the final project goal. The solution will be implemented in three main iterative phases of incremental implementation and release. Each iterative step will enhance the functionality of the system but also update the requirements and system design.

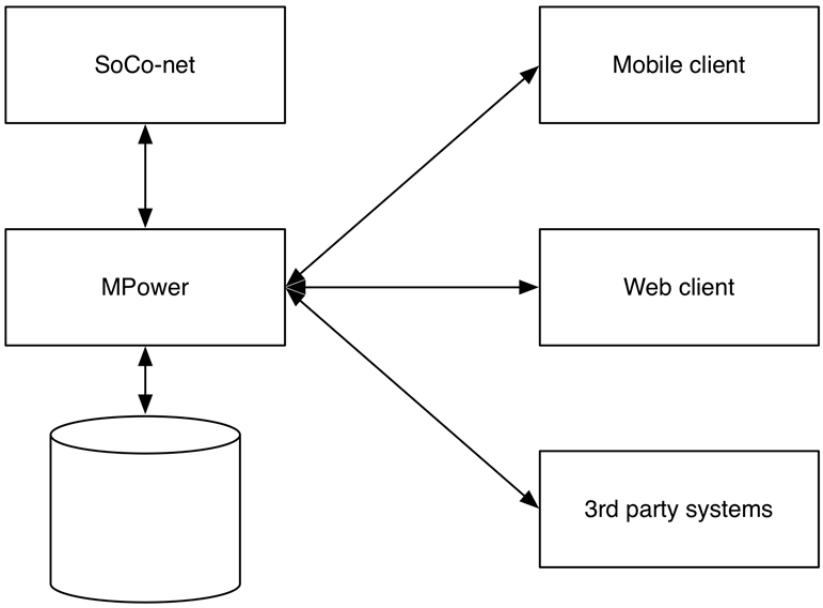

Figure 2: Co-Living high-level component view

Our approach begins by identifying a series of large scale components with definite functionality, essentially decoupled from each other but interacting together to supply the whole functionality of the system. Once the architecture is relatively stable, implementation can begin. The various components will be implemented one at a time supplying basic functionality at the first cycles and refined at later stages. In addition to the implemented components, the solution can make use of "third-party" software as long as it provides the appropriate functionality and implements the appropriate interface. At each implementation stage, feedback from the users will make sure that we do not stray from what they want and also set the priorities for what the following release should contain.

Figure 2 depicts the overall, high-level component architecture of the Co-Living system. SoCo-net component, mobile and web clients, as well as other "third-party" systems interact with the system through the mPower middleware. A more detailed version and discussion related to this diagram follow in subsequent sections of this document.

2.2 Requirements summary

From its inception, Co-Living has had specific objectives regarding the hypotheses it should verify. The primary objective is the validation of how the Social Community network (SoCo-net) concept contributes to assisted living.

Table 2 summarises the most significant set of requirements that characterize Co-Living and that the Co-Living system must address.

As part of its focused approach, the Co-Living PP document [1] is a significant source of (functional) requirements. Other sources include the Co-Living documents including the specification of user socialisation needs [2], the use case scenarios [3] and the security and privacy considerations [4], all from WP1. An exploration of the SoCo-net concept in WP2 provides further requirements.

As a whole, these requirements build the foundation of the design considerations (see Section 3) and the actual design (see Section 4).

Table 2: Initial requirements summary

Req. class	Req. no.	Description	WP
Platform / System	R1	Scale up mPower platform.	PP
	R2	Asynchronous (mobile, messages) communication and notification (SMS, Email, etc.).	1
	R3	Interactive remote (mobile) decision making by end users.	1
SoCo-net	R4	Build and enable the effective management and collaboration of Virtual Care Teams around the elderly for continuous care provision.	1, 2
	R5	Ensure that the elderly will have a unique personalized profile of disabilities and abilities, special needs and preferences promoting thus personalized care provision.	1, 2
	R6	Support the ICT-based services to address the elderly social interaction context categories of Care & Wellness, Guidance and Mobility Monitoring.	PP, 1
SoCo-net Behaviour analysis	R7	Adaptive user profiling techniques, considering user feedback and historical data, to adapt the elderly preferences, capabilities, social relationships and contexts.	PP
	R8	Analysis of participation in social activities.	1
	R9	Analysis of use of Co-Living system.	1
	R10	Inference of extended social network.	1
SoCo-net Education & Feedback	R11	Remote training and intelligent explanation generation systems to help the elderly to make use of the Co-Living services.	PP
	R12	Detect use of Co-Living platform and services and offer personalized instructional material or signal care giver.	1
ICT Services	R13	Voice call, message exchange, and other possible communication means between users.	1
	R14	Capture user confirmation (button).	1
	R15	Care giver (formal, informal) interaction terminal (synchronous and asynchronous).	1
ICT Services Care & Wellness	R16	Monthly plan of physical exercises and activities. Maintain a monthly activity program based on the preferences of the care centre inhabitants.	1
	R17	Creating meeting groups for leisure activities and organize (participation in) group activities with other members.	1

	R18	Competence/knowledge/skills exchange.	1
	R19	Assistance. The user can request help from a caregiver or the caregiver can offer help to the elder.	1
ICT Services Guidance	R20	Daily routine reminder service assists elderly by providing memory help reminders i.e. accessories such as stick, eye-glasses.	1
	R21	Personal calendar, automatically updated with daily activities. Notify user for scheduled activities.	1
	R22	Walking route guidance service.	1
	R23	Medication reminder service.	1
	R24	Weather forecast. Associate weather information with necessary accessories/ activities.	1
ICT Services Mobility monitoring	R25	Real time tracking to get current user outdoor location.	PP (,1)
	R26	Fall detection .Automatic detection and alerting of caregiver.	PP (,1)
	R27	Daily activity follow-up based on predefined plans: sets up daily schedule with various activities with time, place and group members involved, transmitted to the group member that is responsible for the follow-up of the activities, contact the elderly and enquire variations in the schedule (delay or absence from a meeting etc.).	PP
	R28	Indoor Monitoring. Monitor user activity inside the home/care centre.	1
	R29	Physical fitness feedback. Continuous observation of data related to the physical status of a user. Analyse collected data to generate feedback concerning the user's physical status.	1
Privacy & Security	R30	Basic authentication (SSO), confidentiality, integrity, authorization and auditing means.	1
	R31	Assisted (user-friendly, delegation, flexible) access and consent policy management.	1
	R32	Advanced auditing (end user queryable, user-controlled logging, structured logging).	1
	R33	End-to-end access and consent control policy enforcement (obligations, data-centric policies).	1
	R34	Emergency override with auditing.	1
Non-functional requirements			

	R35	User-friendly system interfaces.	PP
	R36	End user adaptable multi-modal interface that operates on any state of the art mobile wireless web browser enabled devices and provides a connection point to wireless sensors and devices associated with contextual parameters assisting the elderly for self-management.	PP

3 Design considerations

This section describes the underlying design considerations for the Co-Living system architecture.

3.1 Concerns

Concerns are related to the documentation of the functional aspects of the Co-Living system and its environment. Functional aspects that are considered to be of such importance that should be treated separately and be specifically visible in the documentation should be identified and treated as a concern. Concerns are related to functionality and may be grouped into two main groups. These are:

- **Application specific functionality (ASF) concerns:** This group covers the functionality that will be necessary to implement if we consider an ideal world with no performance problems, network limitations, system failure and so on. The main ASF concerns for Co-Living are:
    - Information exchange through messaging;
    - User profiles;
    - Security services;
    - Data storage management and persistence;
    - Management of other resources (sensor devices etc.);
    - More concerns will be added for the next version.
- **Quality related functionality (QRF) concerns:** This group covers all types of functionality that may be used to improve the quality of a system according to what is required. For systems with high complexity and high quality requirements a wide range of such concerns must be explicitly defined and handled. The main QRF concerns for Co-Living are:
    - Availability and reliability of solution;
    - User friendliness and ease of use;
    - Standards compliance;
    - More concerns will be added for the next version.

3.2 System assets

System assets are sources of information that can be used when developing the architecture descriptions. System assets can be considered as implicit requirements, which are not necessarily included in the requirement view, however assets may be included in component, deployment and realization views. Examples of assets that are available are:

- **Dictionary:** A dictionary is a reference list of concepts important to a particular model aspect or concern along with discussion and/or definition of their meanings and applications. The core dictionary for Co-Living will be defined for the next version of this document.
- **Standards:** A standard is a formalized model or example developed by a standardization organization or established by general consent.

3.3 Goals and guidelines

Before we start with the description of the Co-Living architecture, we need to explicitly document what have been the goals, guidelines, principles and priorities that dominated the design process.

- **Always keep target users in mind:** Understanding how users will use the system we are designing is critically important. Both the design and development phases are closely coupled with the actual services exposed to the end users. For example many users may suffer impairments or physical limitations that affect their ability to use the system. To address such issues,

and to help developers understand how various roles will interact with the system, all phases and cycles of the project have involve the close cooperation of future end users.

- **Reuse existing components wherever possible:** We have reused, wherever possible, standard and existing components of the successfully developed IST FP6 mPower open source middleware platform. This helped us to avoid unnecessary duplication and speed up development by identifying components in the different parts of the system that are the same or very similar.
- **Define reusable components**: Any Co-Living component can potentially become a reusable asset. The different components of the architecture implement a number of interfaces, each consisting of a set of functions that are specialized for some type of interaction. These interfaces have been defined as generic as possible. That is, all component interfaces are independent of the actual implementation code and also of the concrete data types that will be added by the users when customizing the system.
- **Be flexible:** The goal is to allow Co-Living users to customize the system to meet their unique needs. For example, different users could have access to different sets of services. Therefore the various system components are designed to be as self-supportive as possible and loosely coupled between them. Letting users select options in various components and by letting them modify default values also enhances flexibility.
- **Data centred architectural style:** Since many developers have been collaborating in the Co-Living software development, we have aimed to minimise the coupling between independent components.

3.4 Reference architecture

A reference-architecture is a high level, generic architecture that is used as the basis for development of concrete system architectures, and to compare architectures of existing systems to each other.

The mPower middleware, on which the Co-Living system is build, follows the IBM SOA reference architecture [5] as shown in Figure 8. Consequently, Co-Living also adopts the SOA reference architecture. A brief description of SOA architecture follows in Section 3.4.1 and a detailed description can be found in [6]. Further details on the SOA architectural template of the mPower platform are available in Annex A.

3.4.1 SOA reference architecture

This section briefly describes the main elements of IBM SOA reference architecture and the relationships between them.

IBM [7] defines SOA architectural style as: “A set of patterns and guidelines for creating loosely coupled, business-aligned services that, because of the separation of concerns between description, implementation, and binding, provide unprecedented flexibility in responsiveness to new business threats and opportunities.”

The goal of using SOA is to liberate the business from the constraints of technology without throwing already existing things out. The main advantage of SOA is reusability. User and business needs are constantly changing and requests for new programs just keep coming. In a SOA approach, applications are build using a set of building blocks know as components (some of them are available “off the shelf” and some are built from scratch), so when you need to add new or update existing logic of some application it is not such a big deal with SOA. You only need to change the business logic and the plumbing can stay the same because these two parts are well separated.

The main parties that are involved in SOA are (Figure 3):

- **Service Provider**: Provides the description and implementation of a service;

- **Service Consumer**: Can either use the uniform resource identifier (URI) for the service description directly or can find the service description in a service registry and bind and invoke the service;
- **Service Broker**: Provides and maintains the service registry.

The next diagram shows the main interaction between the above-mentioned parties.

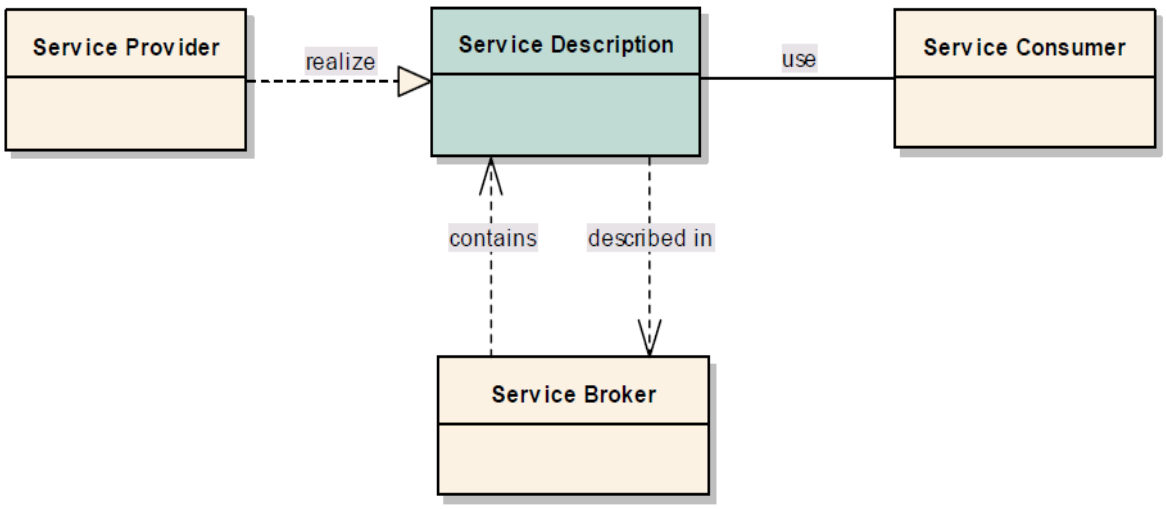

Figure 3: SOA conceptual model

Web services are the preferred standards based way to realize SOA. Figure 4 gives a basic architectural representation of the web services model including:

- Simple Object Access Protocol (SOAP);
- Universal Description, Discovery, and Integration (UDDI);
- Web Services Description Language (WSDL).

As illustrated, the specifications of the basic web services architecture (SOAP, WSDL, and UDDI) support the interaction of a web service consumer with a web service provider and the potential discovery of the web service description. The WSDL has become a standard programming interface to access any application and SOAP a standard interoperability protocol to connect any application to any other. The provider typically publishes a WSDL description of its web service, and the requester accesses the description using a UDDI or other type of registry, and requests the execution of the provider's service by sending a SOAP message to it.

In Co-Living, the standards of WSDL, SOAP and UDDI are used, and followed by many additional web services specifications that define security, reliability, transactions, orchestration, and data management to meet all the user requirements in terms of functionality and quality of service (QoS).

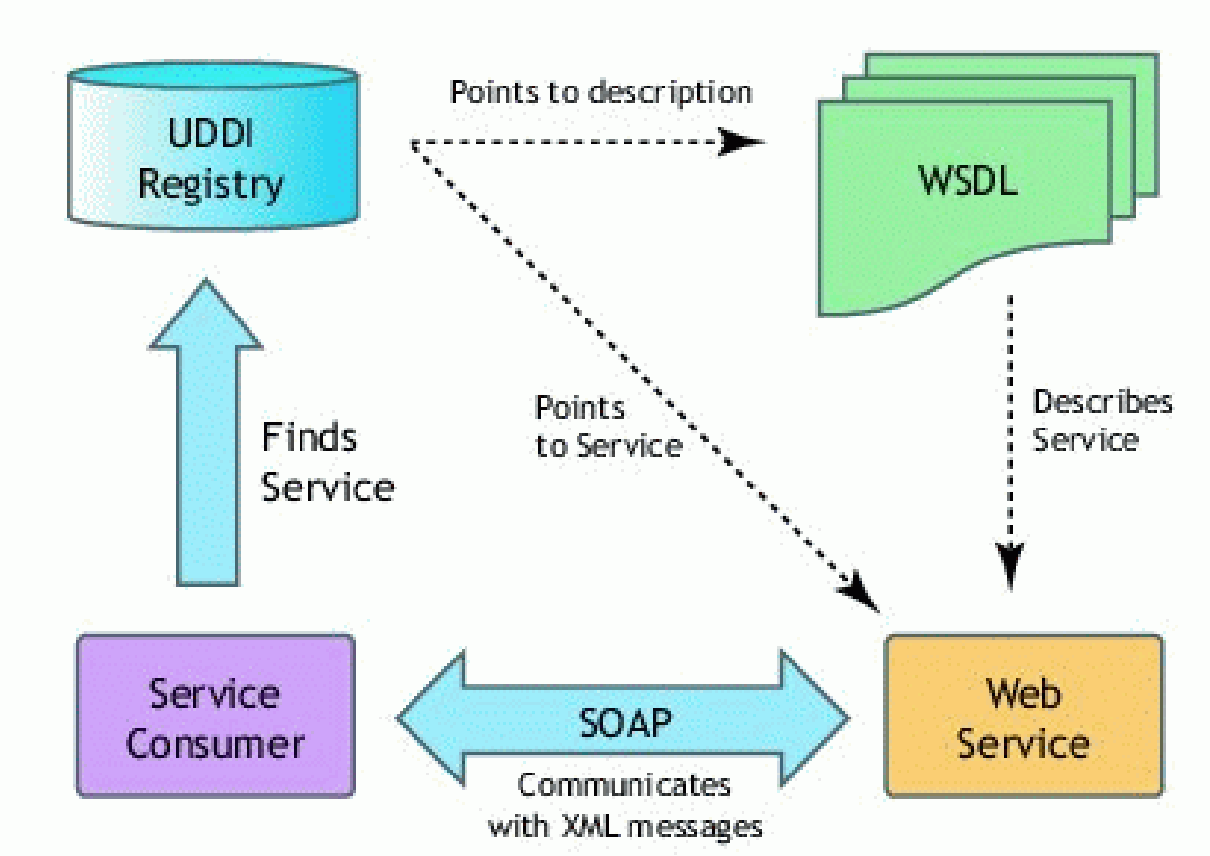

Figure 4: The Web Services model

Using web services helps in overcoming the problem of platform dependence and it also shields user from requiring knowledge of service implementation issues. Furthermore, the service oriented development paradigm is widely adopted due to the following benefits:

- **Reuse:** The ability to create services that are reusable in multiple applications;
- **Efficiency:** The ability to quickly and easily create new services and new applications using a combination of new and old services, along with the ability to focus on the data to be shared rather than the implementation underneath;
- **Loose technology coupling:** The ability to model services independently of their execution environment and create messages that can be sent to any service;
- **Division of responsibility:** The ability to more easily allow business people to concentrate on business issues, technical people to concentrate on technology issues, and for both groups to collaborate using the service contract.

In Annex A, the IBM SOA reference architecture is presented in more detail.

3.5 Development tools and methods

This section describes the method adopted for the architectural design of Co-Living and the major tools and artefacts selected for the design, development, implementation, deployment, and testing of the software system components.

3.5.1 mPower service development methodology

To develop the Co-Living server-side components (web services), the project reuses the development tools and methods from the mPower project. This is a model-driven development approach where each service is modelled in Unified Modelling Language (UML), using a shared information model. A

model to text transformation engine produces application code for the web services (WSDL file) and accompanying documentation (RTF and HTML). The complete process of web services development using the mPower approach is described in detail in [8] and illustrated in Figure 5.

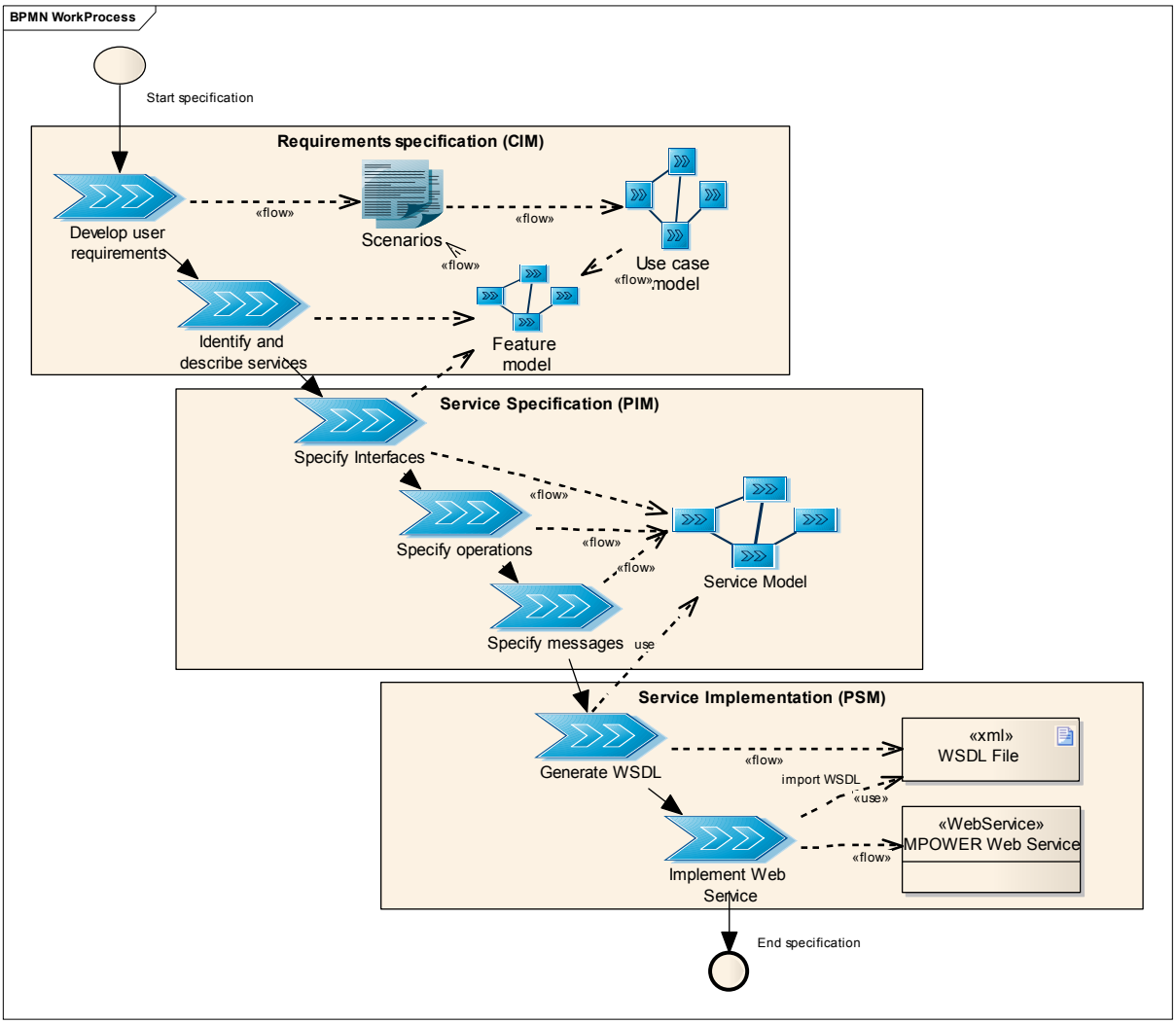

Figure 5: mPower service specification process

3.5.2 mPower service development tool chain

The mPower tool chain for rapid software development is used to support the web service development method. As illustrated in Figure 6, the current implementation of the tool chain is built around various tools and artefacts involved in the development, implementation, deployment, and testing of web services:

- Enterprise Architect for UML modelling and transformations;
- Netbeans for Java development;
- soapUI plugin for Netbeans for testing of messages;
- Glassfish Application Server for deployment of services.

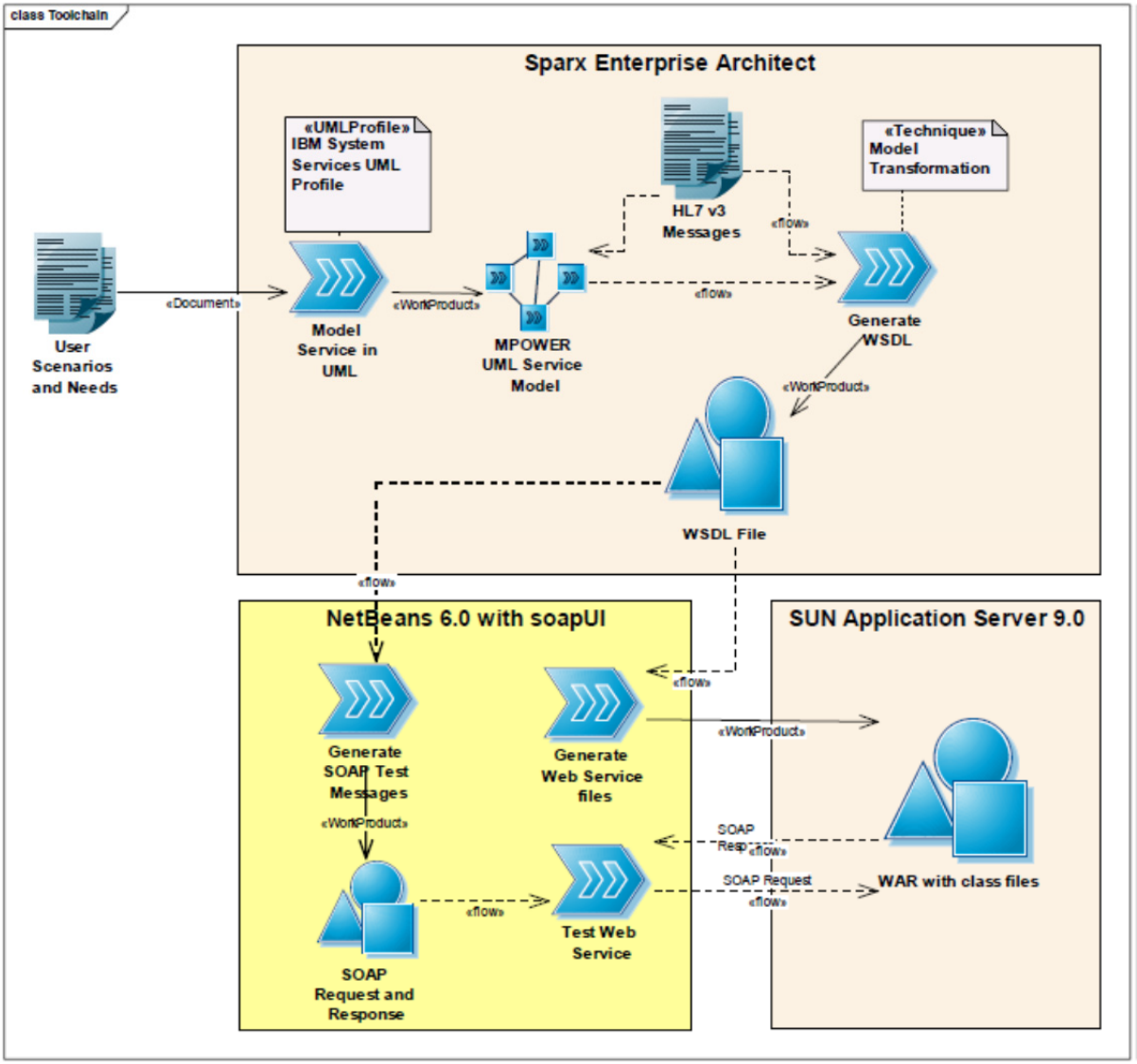

Figure 6: mPower service development tool chain

The Enterprise Architect tool is used to model the services in a UML class model, based on the description of the user scenarios and use case models. Based on the UML model of the services, WSDL models [9] are generated. The WSDL model is further transformed to a WSDL file, which is used as input for the service implementation.

From Netbeans, the WSDL file is imported and is used to generate the web service files that are needed for the implementation of the web service. When the developer has completed the implementation Netbeans is used to build the service and get the WAR file that contains all the class files. This WAR file can be deployed on an application server to make the service available for testing and for use by other services and applications.

More details, on how to set up a development environment and how to use the various tools of the mPower tool chain are provided in [8].

4 System design

This section provides a high level overview of how the functionality and responsibilities of Co-Living were partitioned and assigned to various components. A software component is defined as a part of the system that performs a single function and has a well-defined interface. The main purpose is to gain a general understanding of how and why the system was decomposed, and how the individual parts work together to provide the desired functionality. A subsequent section exists to provide the detailed descriptions of the individual components.

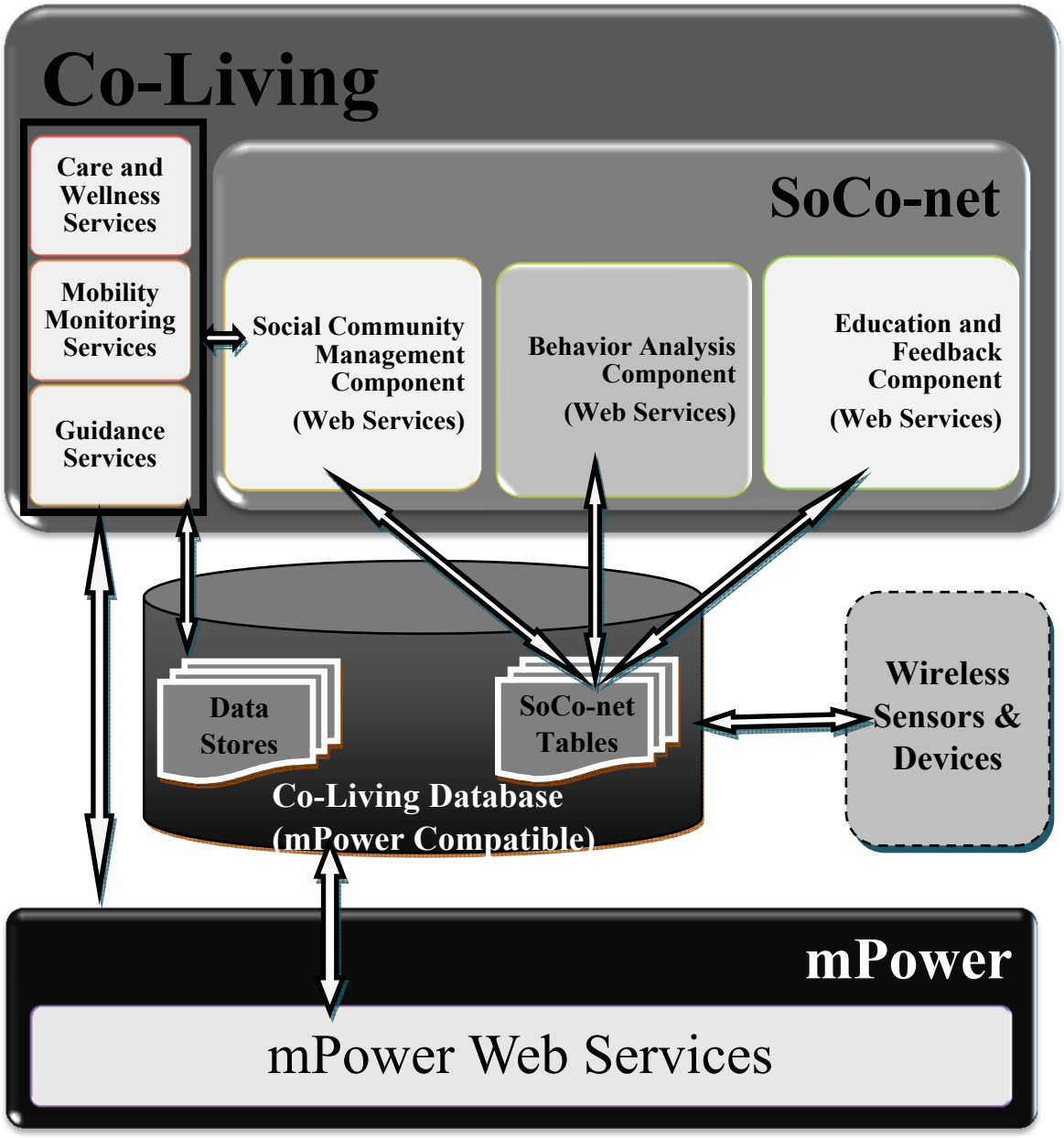

Figure 7: Co-Living decomposition model

4.1 System component view

The purpose of the component view is to describe the system in terms of its subsystems and information objects, and to document how subsystem interaction and information processing is carried out in order to provide the desired behavioural effect. The argumentation for what a specific component shall include should be based on specific criteria.

As part of a component view, models are created with special focus on information, system decomposition, system collaboration, and component and interface specification. The component view is kept at a functional level, not dealing with technology choices. The technologies of choice can have their own preferred approaches and architectural recommendations. Figure 2 in Section 2.1 presents a Co-Living high-level component view. SoCo-net components, mobile and web clients, as well as other "third-party" systems interact through the mPower middleware.

4.2 System decomposition model

The system decomposition model describes how the target system is divided into different subsystems or components, and how these are related to form a coherent whole. A modularised description of the system facilitates reuse of well-defined components and identifies points of integration with other systems.

Decomposition of the system is applied hierarchically, with the initial diagram showing an overview of the whole system (Figure 2). Further decomposition of subsystems is performed when this is seen as part of the architecture; while decomposition of more fine grained components are typically considered as detailed design and not handled in the architecture description.

Figure 7 is a detailed version of the "Co-Living high-level component view" presented in Figure 2. It illustrates in detail the major components that have been identified during the design of the architecture. Next, we briefly present the components depicted in Figure 7 to allow us to explain how they interact to provide the main required functionality of the framework. More detailed descriptions of each component can be found in Chapter 5 of the document.

4.2.1 Components and interactions

The following components have been identified during the design of the Co-Living architecture. The basic functionality of the system can be described in terms of the interactions between them.

mPower middleware

The mPower middleware platform defines and implements an open platform to simplify and speed up the task of developing and deploying services (in a SOA environment) for elderly and persons with cognitive disabilities. The project developed the platform as a suite of independent building blocks supporting:

- Mobile users who often change context and tools.
- Integration of smart house and sensor technology;
- Interoperability between profession and institution specific systems (e.g. Hospital Information System);
- Secure and safe information management, including both social and medical information;

The mPower technical approach defines a model-driven development methodology and a tool chain that supports documentation of system requirements, modelling, design and development of services.
mPower's consortium also offers an open source project including a number of reusable services to be used by developers to rapidly design and develop new applications. The services are related to social and medical information, sensor technology, communication, secure information management, etc.

We have reused, wherever possible, and extended the existing services to avoid unnecessary duplication and speed up the development process.

SoCo-net

SoCo-net is an elderly centric, web based Social Community network, which constitutes a core component of the Co-Living system. Key features of SoCo-net include management of the user profiles and organization of the virtual collaborative network of users grouped in teams around the elderly. Existing mPower services are used by SoCo-net whenever it is applicable, and Web 2.0 technologies (more specifically web services [10]) are used for the development of new services integrated in the SoCo-net component and enhanced the functionality for the virtual community network.

ICT-based services model

Co-Living implements a forest of services, seamlessly combined together to address the various needs and requirements of the users (

Table 2). To support modularity, expandability and flexibility the ICT-based services are grouped into cohesive physical units, which we call packages. Each package contains a set of services that perform functions, which can be anything from simple to complicated processes. The majority of the services have been implemented from scratch however wherever possible existing services, previously implemented under the mPower platform have been reused, extended and tailored to the needs of Co-Living.

Security and privacy model

Personal data is the core resource that the Co-Living services build on. It is crucial for the users to feel that the system does not allow unwanted privacy intrusions and to ensure that they are in control of the services offered to them. Building on the existing mPower security and privacy infrastructure Co-Living designed and developed further security services to realize the required security and privacy functionality and empower elderly with control and transparency over their personal data.

5 Description of components

In Chapter 4, a high-level description of the Co-Living architecture is given; in terms of the major components which have been identified and their interactions. This section presents more information about these components. Namely, each subsection refers to a system software component illustrated in the Co-Living decomposition model of Figure 7.

Along the different project WPs various deliverables are developed to address the design and development of the components presented next, giving more fine-grained details and diagrams showing the component structure, behaviour, or information/control flow. For each subsection where such a document exists, a reference to it is provided.

5.1 mPower

5.1.1 Overall architecture

The overall architecture of mPower is an adaptation of the layered model specified in the IBM SOA Reference Architecture [5]. As depicted in Figure 8, the mPower architecture consists of five main layers and three sidecars. Each layer comprises a set of components that conform to specific rules and requirements [6].

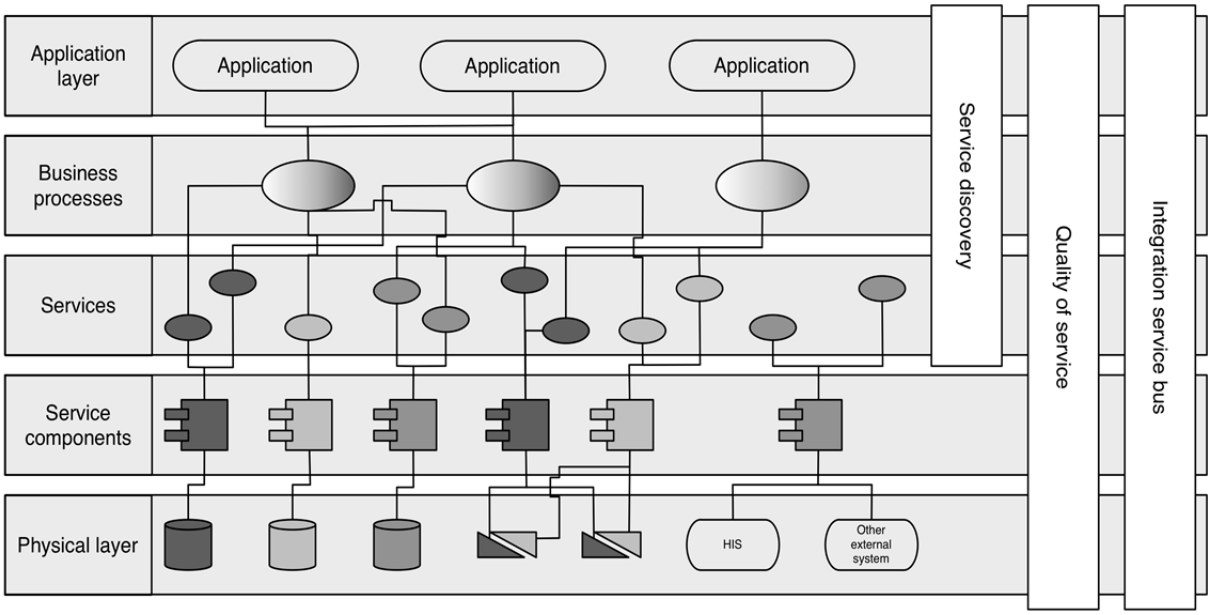

Figure 8: The mPower layer model

Application layer

The application layer of the mPower architecture provides the user interfaces and components that are specific to the application. This ensures that they are segmented from the underlying business processes and services on which they build.

Business processes layer

The business process layer defines the business processes belonging to the applications on layer above. The underlying services are bundled through *orchestration* or *choreography*. Thus, working together to support the use cases and business processes on the application in question. One example of a business process developed in the project is an assistive care process. This business process handles the management of a shared calendar; where the calendar itself, patient and caregiver information, and medical plans are accessed through a set of services and service components.

Services layer

The services layer provides services for the overlying business process layer. Service implementations may use service components and expose their functionality through service interface descriptions. Services can be made available through service discovery by using a registry. The service discovery sidecar handles the service discovery. Examples of services implemented within the mPower project are: user authentication, calendar management, medication management and door control management.

Service components layer

The service component layer contains the different components that comprise the services in the above layer. Service components can also expose any components or databases in the physical layer through interfaces. A typical component in mPower is a smart house sensor driver, which encapsulates and implements the sensor communication logic.

Physical layer

The physical layer consists of databases, existing legacy applications and low-level resources such as sensors and actuators. In mPower one example is the database that contains medication and administrative data or sensors for physiological monitoring and door control.

Service discovery sidecar

The service discovery sidecar support automatically identification of suitable software services that allows satisfying the goal of the requester. In mPower the service discovery is implemented using UDDI. It is a platform independent and XML-based registry. The UDDI specification defines how to publish and discover services through a broker mechanism.

Quality of service sidecar

This sidecar provides the capabilities to monitor, manage and maintain QoS. These capabilities, such as security, performance and availability, are background processes, which by using a sense and response mechanism monitors the health of SOA applications. One important aspect in mPower is security. Security is managed through the security middleware, which ensures sufficient protection for all components on all layers.

Integration service bus sidecar

This sidecar enables the integration of services through the introduction of a reliable set of capabilities, such as intelligent routing and protocol mediation. These capabilities are collectively known as the enterprise service bus (ESB), which provides a location independent mechanism for integration, contrary to the WSDL, which is location dependent. The mPower platform uses the OpenESB platform from Oracle (former Sun Microsystems).

5.1.2 "Free mPower" open source project

The mPower middleware offers an open source project (Free mPower) including various services and components that can serve as a basis for future EU projects and be reused by developers to rapidly design and implement new applications. To enable reuse, these services and components are provided in two different formats: as a UML model or as a compiled component (with source code) ready to be deployed.

The services provided as a part of the mPower middleware are grouped into five categories as illustrated in Figure 9 [11]. A short description about each package of services is provided next and extended details can be found in [11].

- **Management Services**: Services for managing services, users, access rights and system contexts.
- **Information Services**: Services for setting and getting information for individual plan, calendar, medication list, and knowledge sources.
- **Sensor Services**: Services for configuring (add, remove, adjust) devices and retrieving sensor information.
- **Security Services**: Services for authentication and authorization of users and system components.
- **Communication Services**: Services for sending messages and notifications to users and systems.

Moreover, mPower offers other components that are not associated directly with any of those groups of services. An important task for the developers of Co-Living is to review the available assets to extract a reusable set of services and thus avoid starting the ICT-based services design from scratch.

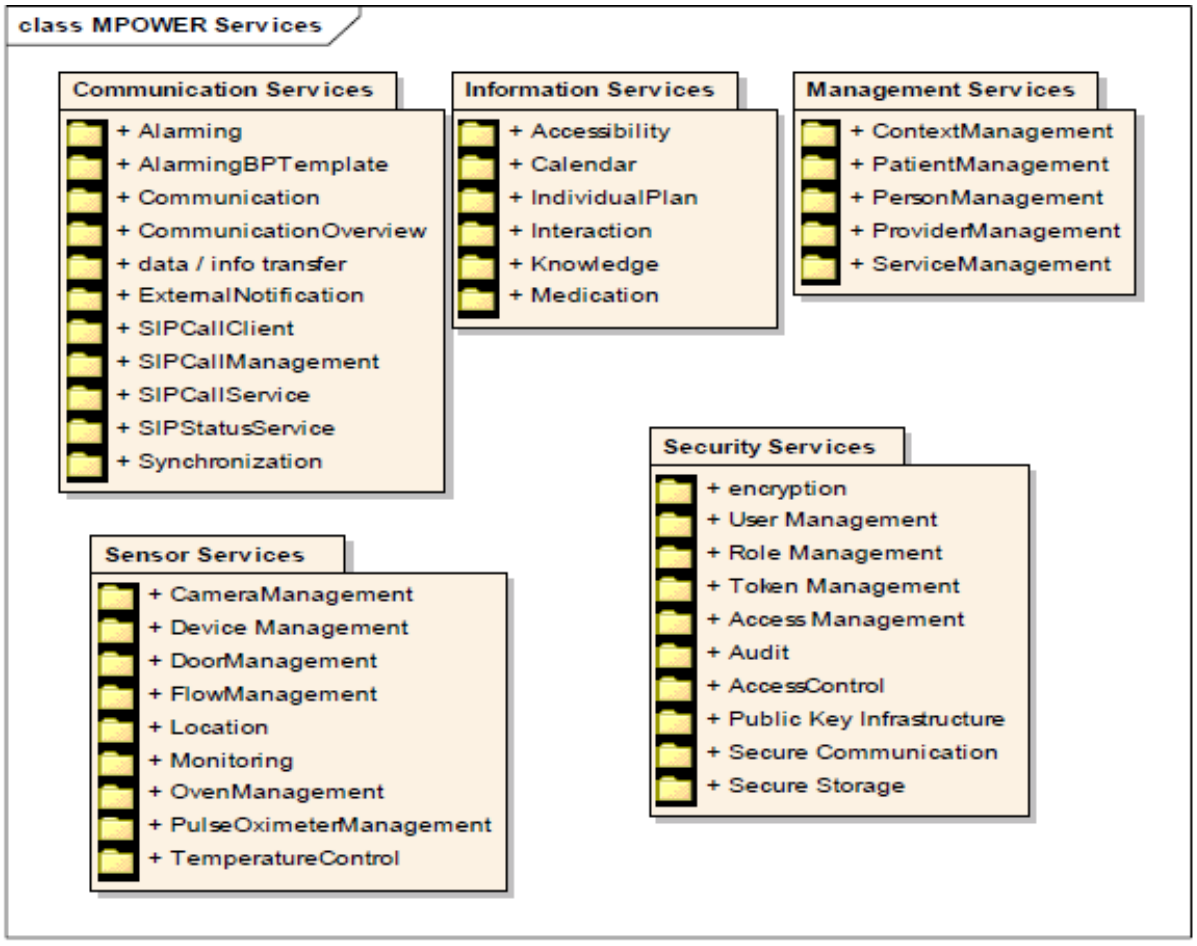

Figure 9: The mPower ICT-based services

5.1.3 Further details

The various deliverables of the mPower project provide extended details regarding the platform. Concretely, the overall architecture and available reusable components and services are presented in [11].

5.2 SoCo-net

SoCo-net (Figure 10), is a Virtual Collaborative Social Community network, which constitutes a core component of the Co-Living system. It is an elderly centric web based network that enables the effective management and collaboration of virtual social care teams around the elderly.

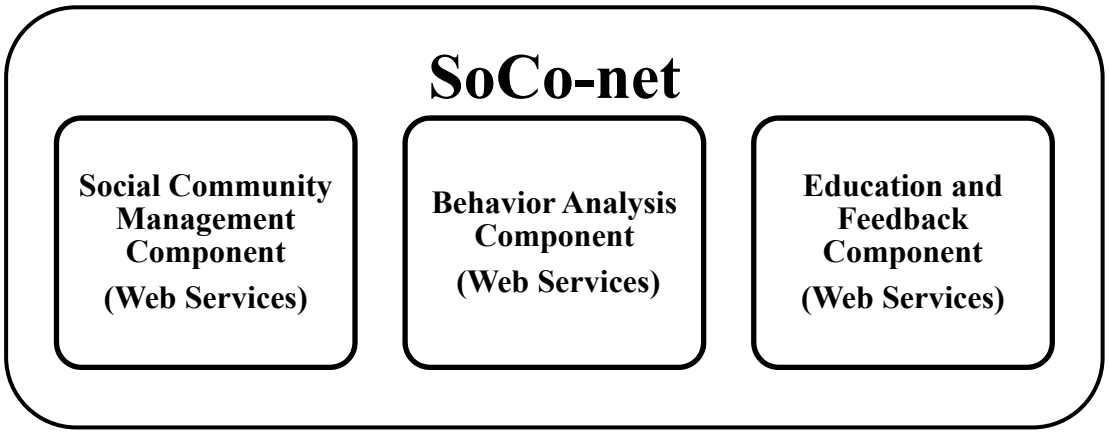

Figure 10: SoCo-net component

As illustrated in Figure 10**,** SoCo-net is made up of three major components integrated together into a coherent module. These are:

- **The Social Community Management component**, which enables the effective administration and coordination of the user profiles and social care teams around the elderly.
- **The Behaviour Analysis component**, to adapt social relationships and contexts of the elderly people as they age. Historical data regarding user behaviour will be used for the identification of changes in the elderly daily activities as he/she ages and trigger actions and related adaptation of the elderly provided services.
- **The Education and Feedback component**, to stimulate the elderly to retain interest in making use of the Co-Living services by the provision of remote training, through intelligent explanation interfaces.

SoCo-net, through the Social Community Management component, builds Virtual Care Teams (VCTs) around the elderly person consisting of people (members) of different ages (young and old) and roles (relatives, friends, neighbours, care professionals, etc.). The members of the VCT will then assist, collaborate and actively communicate with the elderly to improve daily life in an ad hoc and informal way through the use of assistive mobile wireless technologies.

Also, SoCo-net, through the Social Community Management component, ensures that the elderly have a unique personalized profile of disabilities and abilities, special needs and preferences promoting thus personalized care provision. Furthermore, the Social Community Management component of SoCo-net supports, through the web services implemented in this component, different mobile wireless ICT-based services, to address the elderly social interaction context categories of Care & Wellness, Guidance and Mobility Monitoring, by providing information related to the elder's profile and VCTs.

In the Social Community Management component the following four categories of web services are included:

- **Web services for housekeeping management:** These web services are used for adding new items (i.e., activity, accessory, expertise, etc.) in the system, removing items, modifying items or viewing/querying information related to the items.

- **Web services for User management:** These web services are used for adding new users in the system, removing users, modifying users' personal information or viewing/querying users' personal information (e.g., name, address, email, password and telephone number)
- **Web services for VCT management:** These web services are used for creating a new VCT for the elderly, delete a VCT, modify a VCT, or viewing/querying information related to the VCTs.
- **Web services for User Profile management:** These web services are used for creating the profile of the user (e.g. associate the user with certain activities that would like to perform, certain disabilities may have, specific accessories that may need or expertise that may have in a certain domains), deleting the profile of the user, modify the profile of the user or viewing/querying information related to the profile of the user.

SoCo-net gives emphasis on the provision of mechanisms for adapting to changes in user context in a distributed, mobile environment supporting various user contexts. More specifically, in the Behaviour Analysis component, adaptive user profiling techniques and intelligent adaptive interfaces, considering user feedback and historical data (regarding the user behaviour), are used for the identification of changes in the elderly daily activities as he/she ages. By considering these changes, occurring in the elderly behaviour, the elderly profile and preferences are adapted in order to reflect the elderly new habits and way of life.

Finally, based on the elderly profile and preferences, incentives and challenges will be developed, in the Education and Feedback component, to stimulate the elderly to retain interest in making use of the Co-Living services.

5.2.1 Further details

The design of SoCo-net and of its three associated subcomponents, as well as how SoCo-net is integrated with the other components of the Co-Living solution is presented in detail in Co-Living Deliverable 2.2 [12].

5.3 ICT-based services model

In Co-Living, ICT-based services are a category of components that support rapid, low-cost composition of applications offering all the functionality needed by the system, directly or indirectly, and allowing users to interact with it without focusing on underlying technicalities.

The system ICT-based services are grouped into cohesive physical units, which we call packages. Organizing services in terms of packages has several advantages:

- Each package can be owned and authored by a single developer;
- Acceptable dependencies can be specified as part of the overall system design;
- Highly coupled parts of the system can be assigned to a single package, which makes change management easier;
- Packaging aids incremental comprehension, testing and reuse.

Each package contains a set of services addressing a specific user requirement (

Table 2) and performs functions, which can be anything from simple requests to complicated processes.

In accordance to the user needs, documented in WP1, the three major packages of Co-Living services are:

- **Care & Wellness**, providing basic informal care empowering and encouraging the elderly people to undertake physical exercises and activities, create meeting groups for leisure activities and exchange their competence/knowledge/skills with the other members of the virtual social care team.
- **Guidance**, providing instructions, explanation and information supporting the daily activities of the elderly.
- **Mobility Monitoring**, supporting early detection of limitations in mobility and physical fitness and elderly daily activity follow-up based on predefined plans.

Other supportive packages are also identified and form part of the system including among others:

- **Communication Services,** supporting communication of different types such as voice and video call, messaging etc.
- **Security & Privacy Services,** providing well-known security mechanisms addressing legal, ethical, privacy and security requirements.

Services from the different packages cooperate to provide the necessary higher-level functionality of Co-Living. Many of the Co-Living services are implemented using already existing components developed under mPower.

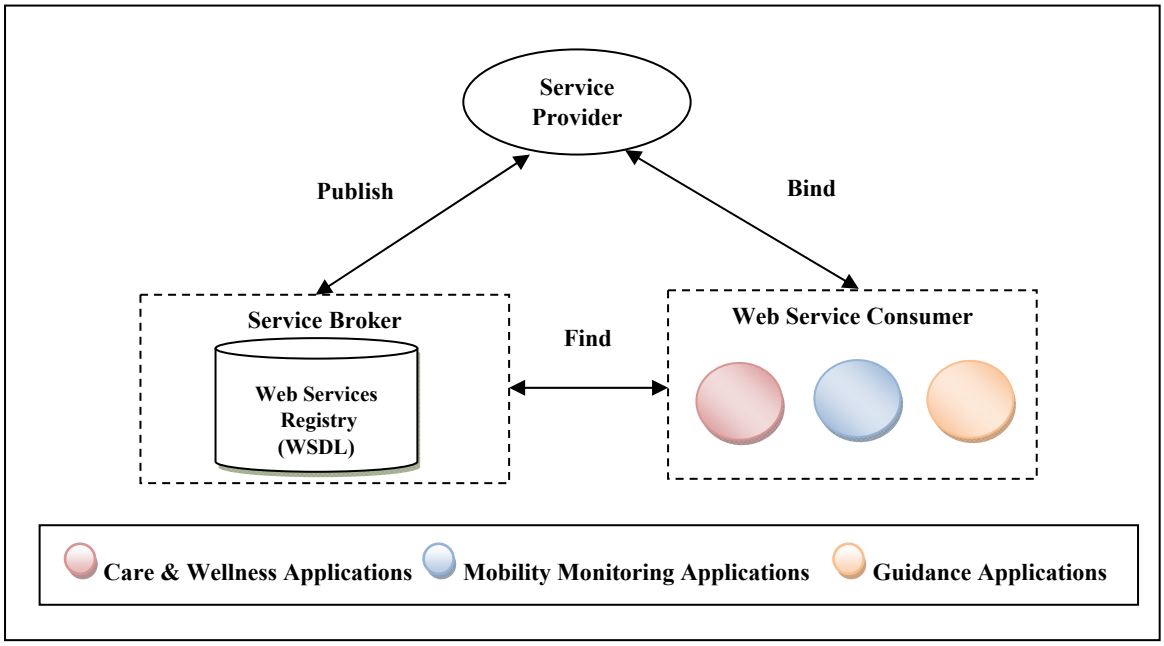

Figure 11: ICT-based service model (SOA)

As illustrated in Figure 11, the Co-Living ICT service model follows the general SOA conceptual model (presented in Figure 3) including service providers and consumers. Service providers are software agents that provide services and are responsible for publishing a description of the services they provide. Service clients are software agents that request the execution of a service and must able to find the descriptions of the services they require and bind to them. Once a web service is bound, it processes the request and sends the response back to the client, through standard interfaces and messaging protocols using basic request-response operations.

5.3.1 Further details

Task 2.3 “Design of ICT-based Services” is responsible for the definition and specification of the different ICT services of Co-Living. The results are documented in detail in [13]. Furthermore, the Deliverables of WP4 provide full definitions and specifications of the services including description of interfaces, operations and semantic content.

5.4 Database and data access model

Data storage is a key component in the Co-Living system framework. All data that is made available to or transferred between the various components must reside within a data store and the architecture has to guarantee that the system’s ICT-based services are able to retrieve and store data objects to these stores. In this subsection we look at the organization of the system’s data stores and how to access data from them.

5.4.1 Data access model

A flexible solution for accessing the data stores is presented by SOA. It is a different approach over the traditional data access models that involve removing the data access logic from the business objects and placing it all in a separate layer known as the Data Access Layer (DAL).

A DAL works as a link between the application presentation layer and the actual data storage layer. It consists of a collection of classes, interfaces and their methods and properties that are used to connect to a database and perform Create, Read, Update and Delete operations.

Figure 12 displays how, using a DAL, the data access logic and application logic are no longer integrated and instead are maintained separate of one another. The presentation layer represents the consumers of the DAL, which could be services or even application processes.

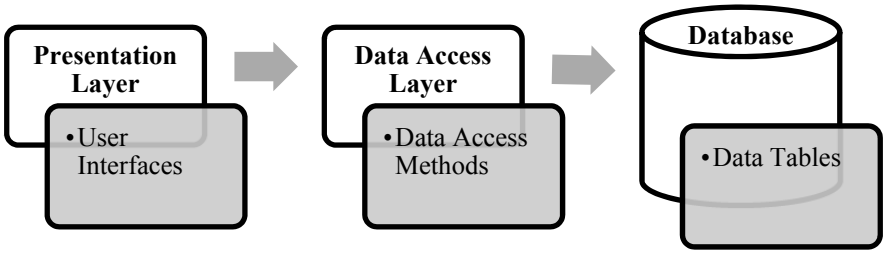

Figure 12: Layered application design

The DAL provides data to the presentation layer objects without using database specific code. This is achieved by exposing a series of data access methods from the DAL that operate on data in the data layer using database specific code but do not expose any database specific method parameters or return types to the presentation layer.

Any time a presentation layer object needs to access data in the database, it uses the method calls in the DAL instead of calling directly down to the data stores. This pushes database specific code into the DAL and makes the object database independent. The existence of a DAL in the application design increases performance, scalability, flexibility and code reuse in the Co-Living services.

For compatibility with the mPower platform, Hibernate is chosen to implement the DAL in Co-Living. This ensures that several data stores can be used as the underlying repository, and the ICT services can access data and store platform related information such as context information and rules independent of any vendor-specific database platform.

Hibernate is a powerful, high performance object/relational persistence and query service that relieves the need to make direct use of a specific Java Database Connectivity (JDBC) API. Hibernate allows to express queries in its own portable SQL extension (HQL), as well as in native SQL, or with an object-oriented Criteria and Example API and it has the ability to generate Java source files to match the structure of a database independent of the implementation platform.

XML files containing configuration data provide Hibernate with details about databases with which it needs to interact. These files contain database connection specifics, connection pooling details, transaction factory settings, as well as references to other XML files that describe tables in the database. Combined, these files provide substantial configurability allowing an application to tune the behaviours and performance of its DAL to a fine level of granularity.

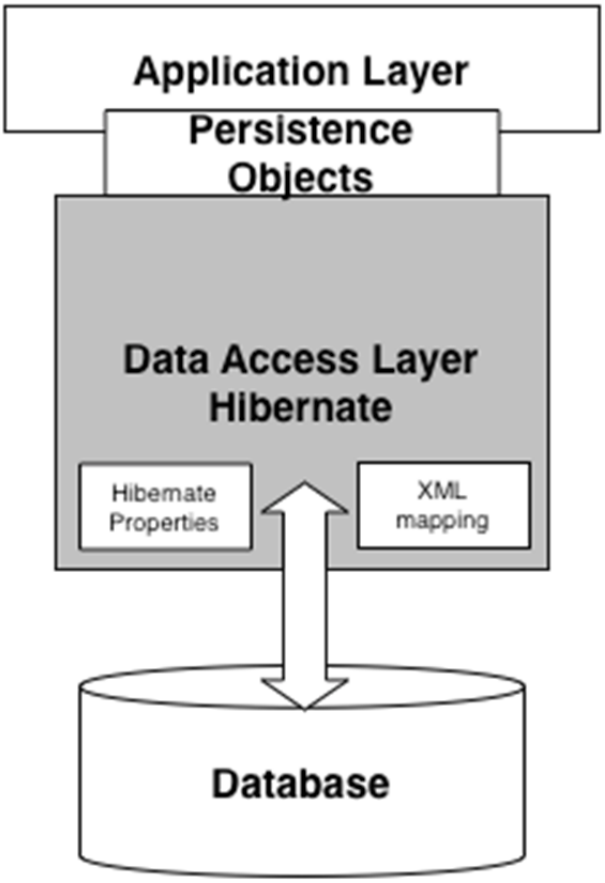

Figure 13: Hibernate architecture

Figure 13 illustrates how Hibernate is using the database and configuration data to provide persistence services and persistent objects to the application. To use Hibernate, it is required to create Java classes that represent the tables in the database and then map the instance variable in the class with the columns in the database. Then Hibernate can be used to perform operations on the database like select, insert, update and delete the records in the table. Hibernate automatically creates the query to perform these operations.

5.4.2 Further details

The SoCo-net data store is described in detail in Deliverable 2.2. The schemas of the Care & Wellness, Guidance and Mobility Monitoring data stores will be specified in detail during the ICT-based Services Development phase in WP4.

5.5 Security and privacy model

Security and privacy is important for Co-Living. Key objective is to protect resources through a set of security controls, both preventive as detective as illustrated by Figure 14.

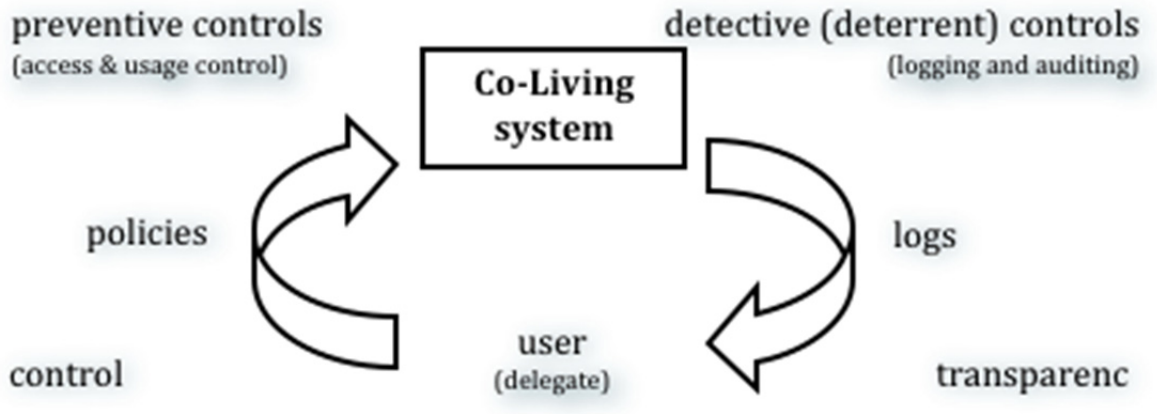

Figure 14: Co-living preventive and detective security controls

The most prominent aspect is that only authorized users may access and use sensitive data or invoke services.

Co-Living scales up the mPower security and privacy functionality in a few aspects that are characteristic for Co-Living: collaborative distributed services, user empowerment and social networking. Co-Living inherits mPower functionality such as authentication, single sign-on and secure communication and certificates as-is. Co-Living focuses on improved support for:

- End-to-end authorization enforcement;
- Centralized logging and auditing;
- Fine grained policies.

One of the typical Co-Living features related to authorization and policies is delegation through its Social Community Network (SoCo-net) approach, which already involves people close to the elderly assisted user whom he can trust. See Figure 15.

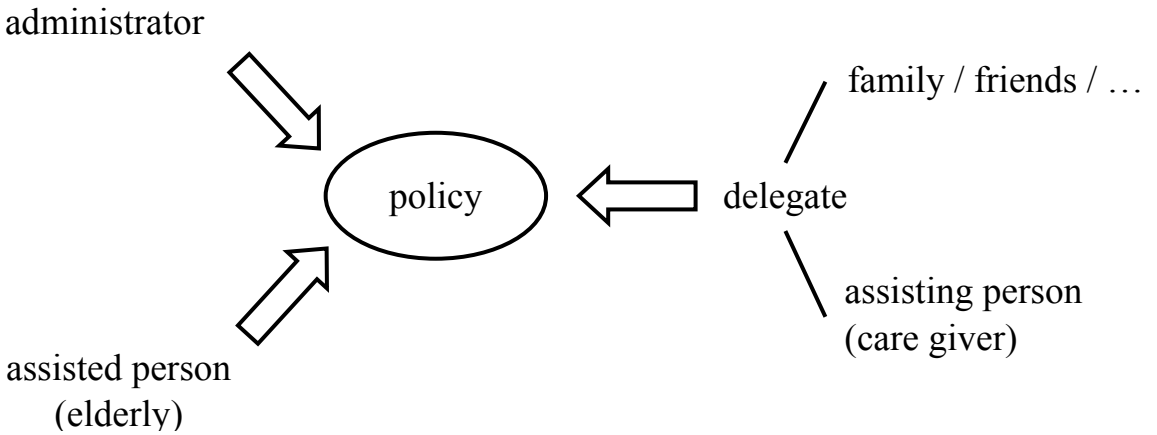

Figure 15: Policy management and user empowerment

5.5.1 Security as a platform and application service

Authorization and auditing security services realize the required security and privacy functionality for Co-Living. The model behind this also explicitly adds security user interaction services to be able to

fulfil its promise that users are truly empowered (if they want to). Concretely, it involves service to manage policies and review audit logs. As common for platform services, security services are orthogonal to application and function-oriented services. Figure 16 illustrates this approach.

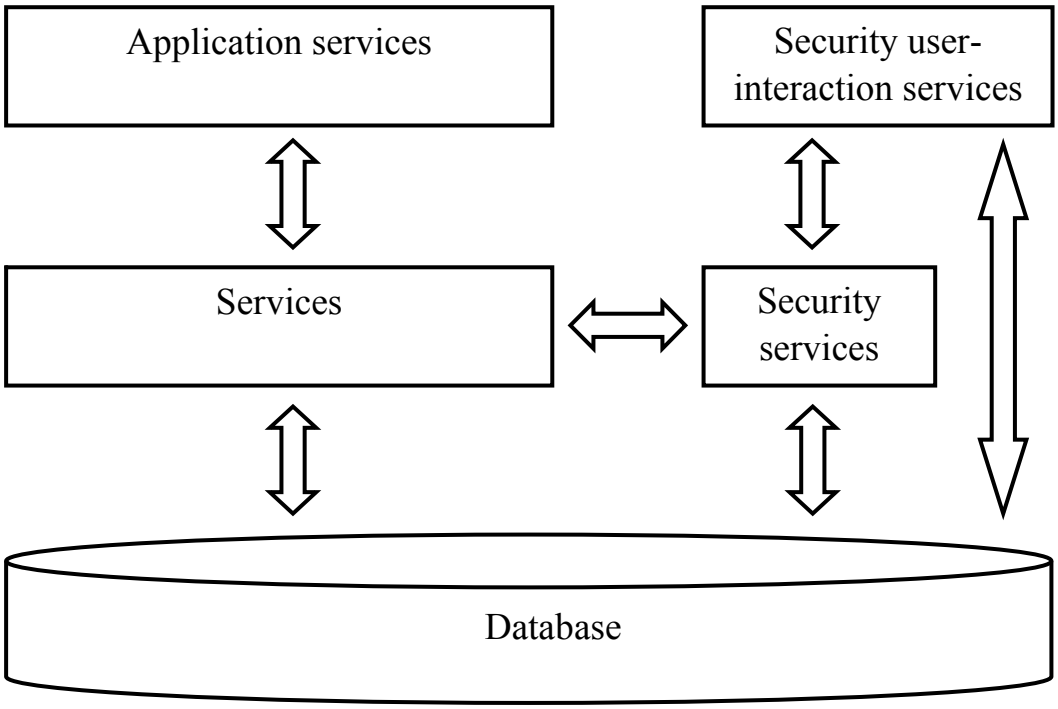

Figure 16: Security in the Co-Living system

5.5.2 Further details

The Co-Living Deliverable 2.2 [12] presents in detail the security and privacy infrastructure concerned with the legal, ethical, privacy and security requirements for the Co-Living system.

6 User interface design

A User Interface (UI) is the component through which the end user interacts with the rest of the system. In general, the various components have interfaces to allow clients to configure and control them. The UI component is the bridge between these internal interfaces and an interface suitable for human interaction.

The UI includes hardware and software components that provide means of input, allowing the users to manipulate a system, and output, allowing the system to indicate the effects of the users' manipulation. More concretely, the UI is typically expected to allow the user to do at least the following:

- Insert or update some data objects;
- Make a selection and view data objects from a data store;
- Make use of a service;
- See and modify the properties of any component of the system.

Any software design that requires the interaction of a human user must follow systematic rules to ensure that the interface exposed to the user is desirable and acceptable. Making the UI of a system easier to use, and matching it more closely to user needs and requirements is the key to achieve specified goals with effectiveness, efficiency and user satisfaction.

This deliverable and its future versions compile a number of guidelines and standards with the purpose to support the developer in the process of designing and creating an accepted UI for the Co-Living application, matching the AAL (Ambient Assisted Living) users' specific requirements.

6.1 Design process

In Co-Living, the UI design process, which results in user interfaces linking all the services together in a federated system, is a structured process. The early stages are paper based, using for example paper mock-ups, and the later stages tend to be based on working prototypes. End users are involved throughout the design and development process.

During phase I of the project the interface design process is only in its initial stages where focus is given on producing a compilation of fundamental principles for designing user interfaces and documentation of common interface standards. These design guidelines are drawn from literature on interface design, especially for elderly users, as well as from personal experience of the partners involved with the pilot sites. The resulting documentation will facilitate greatly the development of interface prototypes and of a stable system.

6.2 Design guidelines

According to Constantine, et al., in their usage-centred design [14], the most important principles to improve the quality of UI design are:

- **The structure principle:** Design should organize the UI purposefully, in meaningful and useful ways based on clear, consistent models that are apparent and recognizable to users, putting related things together and separating unrelated things, differentiating dissimilar things and making similar things resemble one another. The structure principle is concerned with overall UI architecture.
- **The simplicity principle:** The design should make simple, common tasks easy, communicating clearly and simply in the users own language, and providing good shortcuts that are meaningfully related to longer procedures.

- **The visibility principle:** The design should make all needed options and materials for a given task visible without distracting the user with extraneous or redundant information. Good designs do not overwhelm users with alternatives or confuse with unneeded information.
- **The feedback principle:** The design should keep users informed of actions or interpretations, changes of state or condition, and errors or exceptions that are relevant and of interest to the user through clear, concise, and unambiguous language familiar to users.
- **The tolerance principle:** The design should be flexible and tolerant, reducing the cost of mistakes and misuse by allowing undoing and redoing, while also preventing errors wherever possible by tolerating varied inputs and sequences and by interpreting all reasonable actions.
- **The reuse principle:** The design should reuse internal and external components and behaviours, maintaining consistency with purpose rather than merely arbitrary consistency, thus reducing the need for users to rethink and remember.

In Annex B, we present in detail further interface usability good practices especially for the target users of Co-Living. These recommendations are taken from [15].

6.3 Interface standards

In the international ambience, the International Organization of Normalization widely known as ISO, is an international standard-setting body composed of representatives from various national standards organizations with the target to favour the development of the normalization in the world, facilitating the commercial exchanges of services between the different countries.

ISO's portfolio includes several hundred thousand standards and technical regulations of standards. These are available in the ISO Catalogue that can be accessed online [16]. Among these, we have identified **ISO 9241** and **ISO 13407** which provide sets of guidelines related to usability and human centred design of technical systems and products.

The European standard **ISO 9241** [17] defines usability as "the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use". The terms of effectiveness, efficiency and satisfaction are defined as:

- **Effectiveness:** the accuracy and completeness with which users achieve specified goals;
- **Efficiency:** the resources expended in relation to the accuracy and completeness with which users achieve goals;
- **Satisfaction:** freedom from discomfort, and positive attitude to the use of the product;
- **Context of use:** characteristics of the users, tasks and the organizational and physical environments.

The European standard **ISO 13407** [18] defines four activities to design a product from the point of view of the human centred design. These iterative activities, presented in Figure 17, can be used to define, create and evaluate AAL applications.

The process involves iterating until the objectives are satisfied. The sequence in which these are performed and the level of effort and detail that is appropriate varies depending on the design environment and the stage of the design process.

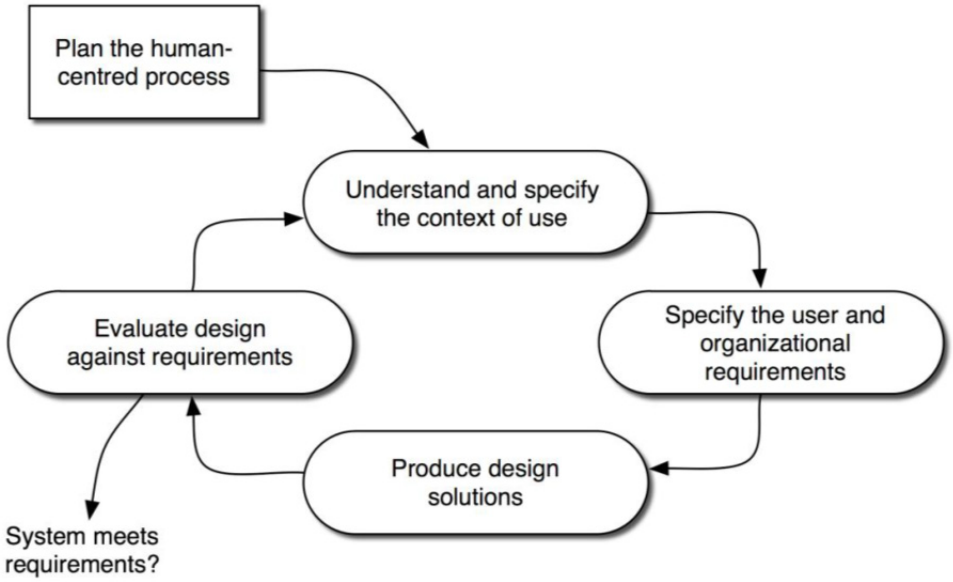

Figure 17: ISO 13407 diagram

1. Understand and specify the context of use. Namely, the physical characteristics of the user, e.g. disabilities and movement impairments; and the environment of use e.g. the user's home, or care centre.
2. Specify the user and organizational requirements. Namely, how quickly a user is able to access wherever he wants with the application.
3. Product Design solution. Introduce the user interaction design in the application.
4. Evaluate Designs against requirements.

Another relevant usability standard is **ISO 9241-10** [19] that defines various principles of dialogue design for interactive systems.

- **Controllability.** The user is able to maintain the focus of a goal over the whole course of the interaction process.
- **Self-descriptiveness.** The interaction should be immediately comprehensible. Feedback to the user is impressible to archive this goal.
- **Conformity with user expectations.** The interfaces should be designed according the user's education, experience, knowledge, etc.
- **Error tolerance.**
- **Suitable for the task.**
- **Suitability for individualization.** The interaction should be modifiable according the user's needs and skills.
- **Suitability for learning**. The interaction provides guidance and support to the user during the learning phase.

By integrating these principles into the four activities depicted in Figure 17, the designers should be able to define intuitive user interfaces for the Co-Living services.

7 Glossary

Table 3: List of terms, abbreviations and acronyms

DAL	Data Access Layer
ESB	Enterprise Service Bus
AAL	Ambient Assisted Living
ASF	Application Specific Functionality
Co-Living	Collaborative Social Living Community
DAL	Data Access Layer
ESB	Enterprise Service Bus
ICT	Information And Communication Technology
PP	Project Plan
QoS	Quality Of Service
QRF	Quality Related Functionality
SOA	Service Oriented Architecture
SoCo-net	Social Community Network
UDDI	Universal Description Discovery And Integration
UI	User Interface
UML	Unified Modelling Language
URI	Uniform Resource Identifier
VCT	Virtual Care Team
WSDL	Web Service Description Language

References

[1] Orbis, Co-Living Proposal (Part B), February 2009.

[2] Orbis, Co-Living Deliverable D1.1: Specification of user socialisation needs and analysis and design of the innovative social practice-oriented community model, May 2011.

[3] Orbis, Co-Living Deliverable D1.2: Specification of use case scenarios, May 2011.

[4] Philips, Co-Living Deliverable D1.3 Specification of Ethical, Privacy and Legal Considerations, January 2011.

[5] A. Arsanjani, "Service oriented Modelling and Architecture: How to Identify, Specify, and Realize Services for Your SOA," 2004.

[6] P. Clements, D. Garlan, L. Bass, D. Garlan, J. Ivers, R. Little, R. Nord, and J. Stafford, Documenting Software Architectures: Views and Beyond, Addison-Wesley Professional, 2002.

[7] E. Gamma, "Design patterns: elements of reusable object-oriented software," 1995.

[8] Sintef, mPower Developers Handbook, November 2008.

[9] World Wide Web Consortium (W3C), Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language, R. Chinnic, et al., Editors. 2007, W3C.

[10] World Wide Web Consortium (W3C), Web Services Architecture, D. Booth, et al., Editors. 2004, W3C.

[11] Sintef, mPower Deliverable D1.1: Overall Architecture, mPower consortium, November 2008.

[12] IPN, Co-Living Deliverable D2.2 : Design of SoCo-net and of a security and privacy Infrastructure, May 2011.

[13] IPN, Co-Living Deliverable D2.3: Design of ICT-based services, May 2011.

[14] L.L. Constantine and L.A.D. Lockwood, "Software for use: a practical guide to the models and methods of usage-centred design," Apr. 1999.

[15] Sintef, mPower Deliverable D8.2: Socio-economic, regulatory and policy studies, mPower consortium, November 2008.

[16] ISO - International Organization for Standardization. Intec. Retrieved May 28, 2011, from http://www.iso.org/iso/home.htm.

[17] ISO/IEC 9241-14 Ergonomic requirements for office work with visual display terminals (VDT)S-Part 14 Menu dialogues ISO/IEC 9241-14:1998(e),1998.

[18] ISO/IEC 13407 Human-Centred Design Processes for Interactive Systems, ISO/IEC 13407; 1999 (e), 1999.

[19] ISO/IEC 9241-10 Ergonomic requirements for office work with visual display terminals (VDTs) - Part 10 Dialogue principles ISO/IEC 9241;1996(e),1996.

Annex A IBM SOA reference architecture

A.1 An architectural template for SOA

The IBM Reference architecture defines seven layers. They are presented on the next picture and a description of the each layer is given below.

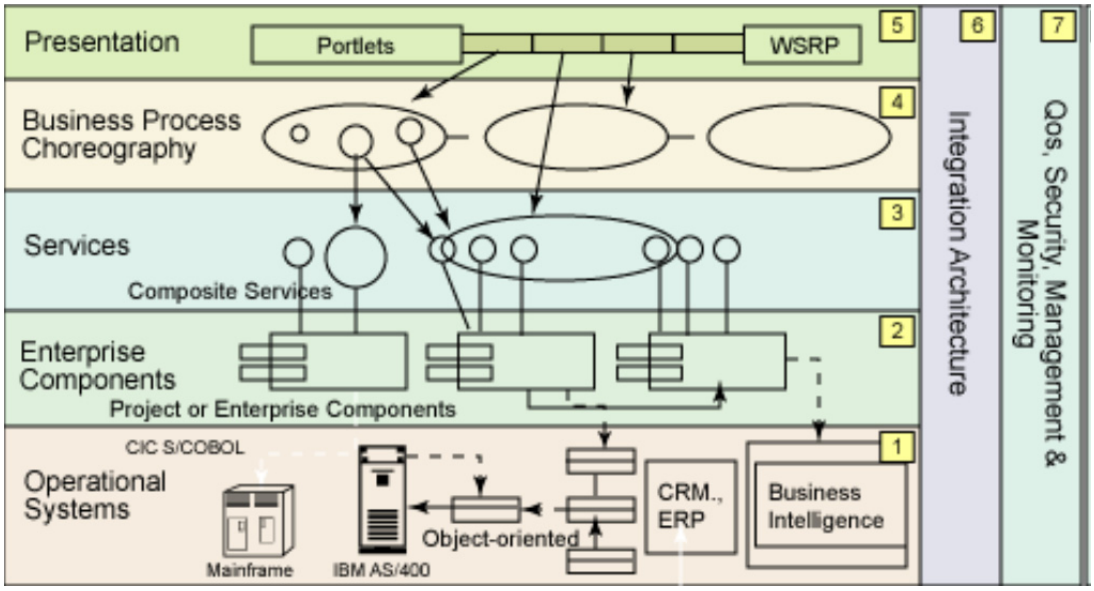

Figure 18: IBM SOA reference architecture

Layer 1: Operational systems layer.

This consists of existing custom built applications, otherwise called legacy systems, including existing CRM and ERP packaged applications, and older object-oriented system implementations, as well as business intelligence applications. The composite layered architecture of SOA can leverage existing systems and integrate them using service oriented integration techniques.

Layer 2: Enterprise components layer.

This is the layer of enterprise components that are responsible for realizing functionality and maintaining the QoS of the exposed services. This layer typically uses container-based technologies such as application servers to implement the components, workload management, high-availability, and load balancing.

Layer 3: Services layer.

The services the business chooses to fund and expose reside in this layer. They can be discovered or be statically bound and then invoked, or possibly, choreographed into a composite service.

Level 4: Business process composition or choreography layer.

Compositions and choreographies of services exposed in Layer 3 are defined in this layer. Services are bundled into a flow through orchestration or choreography, and thus act together as a single application. These applications support specific use cases and business processes.

Layer 5: Access or presentation layer.

This layer is usually out of scope for discussions around a SOA.

Level 6: Integration (ESB) layer.

This layer enables the integration of services through the introduction of a reliable set of capabilities, such as intelligent routing, protocol mediation, and other transformation mechanisms, often described as the ESB. WSDL specifies a binding, which implies a location where the service is provided. On the other hand, an ESB provides a location independent mechanism for integration.

Level 7: QoS layer.

This layer provides the capabilities required to monitor, manage, and maintain QoS such as security, performance, and availability. This is a background process through sense-and-respond mechanisms and tools that monitor the health of SOA applications, including the all important standards implementations of WS-Management and other relevant protocols and standards that implement quality of service for a SOA.

A.2 IBM SOA Architectural template and mPower platform

Following IBM SOA Architectural template the layers 2, 3, 6 and 7 are in the focus of the mPower platform. As the platform is build from scratch there are no existing legacy systems that need to be wrapped but however there are different sensors and actuators that need to be exposed as services independent of their vendor, type or communication standard and extern systems. A new layer called "Physical layer" is added to fulfil these needs.

The mPower architecture that follows the IBM SOA reference architecture is presented in Figure 19.

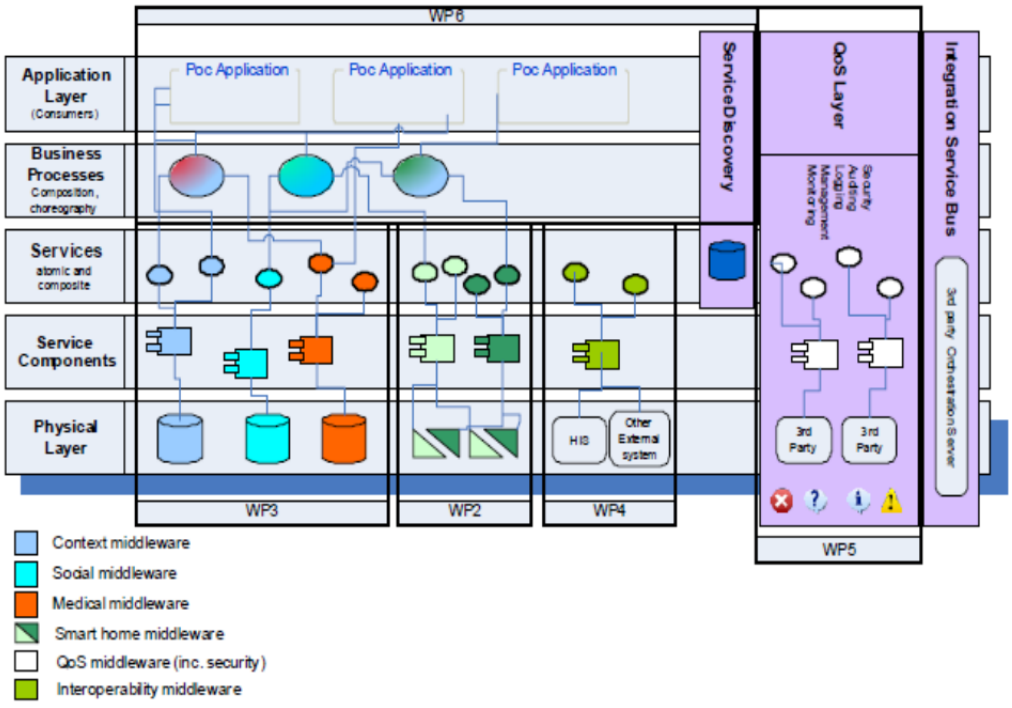

Figure 19: mPower reference architecture

Annex B Usability good practices for the target end users

Design adjustments according to the limitations related to ageing are the focus of most guidelines. Numerous recommendations on website design are published because of widespread usage of web based services. Web usability guidelines usually present more or less the same guidance, which can be summarized as following:

Website layout and style elements:

- Colours – while selecting colours for design it is necessary to avoid extremely bright colours that can be annoying or tiring, to maintain a distinction between colours of the text and background (light background and dark text are more relevant for the elderly users), balancing these considerations with colour-blindness concerns.
- Text – larger font size is preferable, easy to read typefaces should be applied, and more space between paragraph and sections should be provided. When choosing the typeface and letter intensity it is important to note that some fonts become less readable when made bold or italic. Left justification of the text provides better readability, while centred text should be used only in case of titles. Full justification that creates large spaces between words reduces readability. In older age peripheral vision decreases and therefore for the elderly double -spacing is preferable.
- Navigation – one of the important components for navigation is the menu that should be accessible for persons with limited ability for precise movement or with impaired vision. Menu sections are convenient if they are static (avoiding drop-down menus) and large enough to read. Error messages such as “page not found” may be frustrating. Therefore it is important to avoid multiple linking, or to provide more links to the home, or top level, page.
- Frames and tables are not readable by all browsers and can prevent the use of assistive technology for visually impaired persons thus making a website inaccessible. Therefore, some experts argue that an additional version without frames and tables should be provided. Others would make the case for universal design.
- Multimedia – animation elements could be distracting and even decisive element for non-use of the website in case of visual impairments. However, realistic illustrations that accompany text, or directions for e.g. filling forms or other actions, are helpful to assist users to comprehend the information. Illustration should have a textual description allowing assistive technology to recognize it for visually impaired users. When inserting audio files the problem of different additional software for playing it should be considered.

Content organization:

- Memory cues – it is important to place some cues that will help the user to get oriented during the navigation. Site maps that structure the content provided can be a helpful orientation tool.
- Language – clear, non-ambiguous language facilitates using the website. Professional or technical slang should be avoided. Summaries of the content at the top of the page are preferable to make browsing more effective.
- Information arrangement – long documents should be avoided or structured to shorter sections. Clear content organization and avoidance of information and graphic overload facilitates retrieval of necessary information and reduces frustration.

- Help information should encompass both content and technical assistance. Providing alternatives for getting help information as, for instance, email, telephone or frequently asked questions, is important.
- Information about errors should provide some explanation of why the error occurred. Search engine hints that explain the system logic and suggest some recommendations on search improvement may be crucial.
- Designers collaborating with older adults in the development process of ICT-based technology should be aware of certain peculiarities of such partnerships. Usually elderly people assign all failures of using the system to their own lack of skills or knowledge instead of assuming that the technology is at fault. Even if the system works badly they tend to find some positive aspects and rarely disclose negative ones. However, they will not use it again. Usability tests conducted by AARP Foundation (USA), which supports the elderly, reveal the common denial of being old among the elderly persons.
- Invited to participate in usability test they assumed that it was performed to assess usability for their parents or older friends. Members of usability teams should also consider adult children of the elderly participants of usability projects as usually they assist their parents with ICT and can provide valuable information about problems encountered.