Project Identification	
Project number	No. 611421
Duration	1st Dec 2013 – 30th Nov 2016
Coordinator	Andreas Hochgatterer
Coordinator Organisation	AIT Austrian Institute of Technology GmbH, Austria
Website	www.miraculous-life.eu

Miraculous-Life

Miraculous-Life for Elderly Independent Living

Document Identification	
Deliverable ID:	D5.1b Specification of overall system architecture and security and privacy infrastructure
Release number/date	V09 / 22.12.2015
Checked and released by	Andreas Hochgatterer (AIT)
Work Status	Finished
Review Status	Accepted

Key Information from "Description of Work"	
Deliverable Description	The deliverable describes the Miraculous-Life architecture and its components. It also described basic technologies used as well as the relation of the stakeholder groups and artefacts to the architecture. The deliverable explains also the security and privacy mechanisms and components integrated in the Miraculous-Life system and architecture.
Dissemination Level	PU=Public
Deliverable Type	R = Report
Original due date	Project Month 24 / 30. November 2015

Authorship& Reviewer Information	
Editor	Sten Hanke, Emanuel Sandner / AIT
Partners contributing	Citard, Noldus, UCY, UniGe

Reviewed by	Maher Ben Moussa (UniGe)

Release History

Release Number	*Date*	*Author(s)*	*Release description /changes made* *Please make sure that the text you enter here is a brief summary of what was actually changed; do not just repeat information from the other columns.*
V01	23.04.2015	SH/AIT	First version of version b – structure new content according to review results
V02	26.06.2015	SH/AIT	First content for security of KB and CoNet
V03	18.11.2015	CC/Citard	Update of CoNet security implementation
V04	02.12.2015	SH/AIT	SOA references and chapter
V05	05.12.2015	SH/AIT	Overview of ML components and clarification
V06	10.12.2015	ES/AIT	Security update
V07	20.12.2015	SH/AIT	Final updates and fine tuning
V08	21.12.2015	MBM/UniGe	Internally Reviewed
V09	22.12.2015	SH/AIT, ES/AIT	Review update, finalization

Miraculous-Life Consortium

Miraculous-Life (Contract No. 611421) is a project within the 7th Framework Programme. The consortium members are:

Partner 1	***AIT AUSTRIAN INSTITUTE OF TECHNOLOGY GMBH (AIT, Project Coordinator, AT)***
Contact person:	Andreas Hochgatterer
Email:	andreas.hochgatterer@ait.ac.at
Partner 2:	***UNIVERSITY OF GENEVA (UniGe, CH)***
Contact person:	Maher Ben Moussa
Email:	maher.benmoussa@unige.ch
Partner 3:	***UNIVERSITY OF CYPRUS (UCY, CY)***
Contact person:	George Samaras
Email:	cssamara@cs.ucy.ac.cy
Partner 4	***ORBIS MEDISCH EN ZORGCONCERN (ORBIS, NL)***
Contact person:	Cindy Wings
Email:	c.wings@orbisconcern.nl
Partner 5	***FRAUNHOFER IGD (Fh-IGD, DE)***
Contact person:	Andreas Braun
Email:	andreas.braun@igd.fraunhofer.de
Partner 6	***Noldus Information Technology BV (Noldus, NL)***
Contact person:	Ben Loke
Email:	b.loke@noldus.nl
Partner 7	***CITARD SERVICES LTD (Citard, CY)***
Contact person:	Eleni Christodoulou
Email:	eleni_christodoulou@cytanet.com.cy
Partner 8	***ZOOBE MESSAGE ENTERTAINMENT GMBH (Zoobe, DE)***
Contact person:	Sascha Fagel
Email:	fagel@zoobe.com
Partner 9	***MAISON DE RETRAITE DU PETIT-SACONNEX (MRPS, CH)***
Contact person:	Donato Cereghetti
Email:	donato.cereghetti@hotmail.com

Table of Contents

Table of Figures

List of Tables

Abbreviations

Abbrev.	*Description*
AAL	Ambient Assisted Living
ECA	Embodied Conversational Agent
SOA	Service Oriented Architecture
NCF	Noldus Communication Framework
VSP	Virtual Support Partner
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
WS	WebSocket
WSS	WebSocket Secure
HOMER	HOMe Event Recognition system
AMQP	Advanced Message Queuing Protocol

Executive Summary

A core deliverable of the Miraculous-Life project is the overall system architecture and security and privacy infrastructure. The deliverable gives an overview of the overall system architecture, the different modules, the communication interfaces as well as the software distribution. The deliverable also discusses the infrastructure integrated which provides a secure communication and a safe data as well as user handling. This involves also basic functionality for the data protection.

The main components of the Miraculous-Life system are:

- User Recognition Modules
- Speech Recognition Modules
- Environment Context Analysis
- VSP Multimodal Dialogue Management Framework and Avatar
- Knowledge Base
- Home Daily Activity and Safety Services
- Sensor and Sensor connection modules
- Audio Emotion Recognition

The service categories of the Miraculous-Life system are:

- Safety Services (High level environment context analysis, fall detection, dangerous object adviser, obstacle avoidance, emergency situation guidance, risk avoidance, low level safety service)
- Education & Leisure Services (appointment reminder, event creator)
- Guidance Services (motivation for physical activity, object locating)
- Care & Wellness Services (object location reminder, medication reminder)
- General Services (reminder service, alert service, notification service, update WebUI)

The deliverable provides also the overview of the State-of-the-Art and tries to make clear why the different components and technologies of the architecture have been chosen. The deliverable tries to point out which components have been developed inside the Miraculous-Life project and which are third party components used by the Miraculous-Life project.

1 About this Document

1.1 Role of the deliverable

The purpose of this document is to specify the Miraculous-Life overall architecture. This architecture describes all architectural aspects of the Miraculous-Life -based information system. A clear structure of the architecture as well as the use of standard interfaces are important, because they provide the possibility to reuse parts of the system or extend the architecture later on.

The document should enable architects and developers to

- Understand which type of system the Miraculous-Life is
- Why the Miraculous-Life system design has been designed like it is designed now
- Define which actors are involved in the interaction with the Miraculous-Life system
- Document the rationale for the system building blocks based on the user scenarios, use cases and features
- Understand Miraculous-Life Reference Architecture
- Understand how to deploy and run Miraculous-Life services

To accomplish the defined purposes, the document describes not only the proposed architecture but also briefly the process of applying it in the design and development processes.

The primary audience for this document is system architects and developers that

- Deploy the Miraculous-Life system in the elderly homes and residential houses
- Will apply the Miraculous-Life system in the design and / or development of a software system
- Will specify or update the Miraculous-Life system
- Need to understand the architecture of a Miraculous-Life system

1.2 Relationship to other Miraculous-Life deliverables

The deliverable is related to the following Miraculous-Life deliverables:

Table 1: Relationship to other deliverables

Deliv:	*Relation*
D4.1	Design and specification of ICT-based services and safety services
D5.3	Specification of the Miraculous-Life system integration
D5.4	System integration validation
D1.1	Specification of user needs analysis and design of VSP model
D2.1	Specification of the user emotion recognition component
D1.3	Ethical, Privacy, Legal Considerations and Deontological practice

1.3 Structure of this document

The document first explains its role and its connection to the overall Miraculous-Life project. To make it more clear to the reviewer or other interested readers of the documents it clearly explains as well what changes to the document have been applied to the document after the version A release.

These changes include updates on the one side but also new content and chapters to make the document more valuable and better to read.

The next chapter explains the three main considerations of the Miraculous-Life architecture. Basically these considerations follow the projects objectives – to design and system to interact through an avatar, as so called embodiment of the system, to provide services to the elderly user to use the system in a useful way, to integrate components and sensors in an open and modular manner.

The Miraculous-Life system follows certain requirements. Requirements which are coming first from the Description of Work and second from the end user requirements form WP1 in this project. The document explains how these requirements have been addressed in the system design.

The document gives furthermore a complete overview and detailed explanation of all technologies chosen. This includes first all software tools used for developing and running the system and second all hardware components used.

Based on this all Miraculous-Life system components are explained, including the communication protocol as well as the physical deployment.

Major infrastructure components and communication tools in Miraculous-Life like the HOMER (Home Event Recognition System) as well as the Noldus communication framework (NCF) are explained in detail.

To make more clear which components have to be developed or further developed in the Miraculous-Life project, the document provides a chapter with a component overview.

This deliverables as well is explaining all security and privacy considerations and concludes with a conclusion.

Annex A provides more detailed information on the SOA architecture specification.

2 Changes to previous deliverable version

Table 2 provides an overview of the content changes of this deliverable version to version D5.1a.

Table 2: Deliverable changes to previous deliverable version

Deliverable changes	
Action	**Title**
Update	Section “Executive Summary” by adding the new content
Update	Section 1 “About this document” with updates regarding the new content and reviewer recommendation. Especially “Structure of the document”
Update	Section 5 “Miraculous-Life technological overview”. The whole chapter has been updated according to the changes and adaption regarding the technology chosen
Update	Section 6 “Miraculous-Life architecture”. The whole chapter has been updated with minor things regarding the change in the architecture. Basically on the components.
Update	Section 9 “Miraculous-Life services”. This chapter has been updated and gives a short overview of the ML services.
Update	Section 11”Security and Privacy Infrastructure consideration”. This chapter has been updated according the implementation of the security and privacy components in ML.
Update	Section 12 “Conclusion” this chapter has been updated to conclude the deliverable.
Add	Section 2 “Changes to previous deliverable version”. This chapter has been added to better show the changes regarding the previous version of this deliverable. This has been explicitly requested by the reviewers.
Add	Section 3 “Considerations when designing the Miraculous-Life architecture”. This chapter has been added to better reflect the objectives the Miraculous-Life system is embedded in. In reflects also the state-of-the-art regarding the different objectives of the Miraculous-Life system. It serves also as an explanation of why the current architecture has been chosen. This chapter has been explicitly requested by the reviewers.
Add	Section 4 “Miraculous-Life system requirements”. This chapter has been added to explain the architecture decisions based on the requirements from the Description of work as well as the user requirements from the project. This chapter has been explicitly requested by the reviewers.
Add	Section 4.1 “The Miraculous-Life Service-oriented architecture (SOA) approach”. This chapter has been added to explicitly explain this design aspect of the architecture

Add	Section 10 “Overview of the Miraculous-Life components and artefacts”. This chapter has been added to give an overview of all components and explicitly explain which ones have been developed or used in the Miraculous-Life project. This chapter has been explicitly requested by the reviewers.
Add	Annex A to better explain the SOA approach

3 Consideration when designing the Miraculous-Life System Architecture

3.1 Embodied Agent

Human like computer-animated characters, also known as Embodied Conversational Agents (ECAs) [6], have attracted a lot of attention over the past years in the field of artificial intelligence [7]. They are designed to simulate human face-to face conversation with their users and are typically represented as humans or animals, specifically lifelike and believable in the way they behave. Cassell [6] defines ECAs as those virtual characters that have the same properties as humans in face-to-face conversation, including:

- The ability to recognize and respond to verbal and non-verbal input.
- The ability to generate verbal and non-verbal output such as mouth movements, eye movements, head movements, hand gestures, facial expressions, and body posture.
- The ability to deal with conversational functions such as turn taking, feedback, and repair mechanisms
- The ability to give signals that indicate the state of the conversation, as well as to contribute new propositions to the discourse.

Empirical studies [9] reveal that ECAs can improve the natural interaction with elderly users in ambient intelligent environments. Specifically, older adults both with and without cognitive impairment, are capable of recognizing emotions in the facial expressions of an agent and follow instructions much better when interacting with an agent. Another study, concluded that embodied agents allow the development of affinitive relationships with their human partners and can therefore help to fulfil the need of affiliation in ambient assisted living environments [10]. A number of recent systems based on ECAs aim to address the needs of older adults, to provide companionship and assist older adults in health related domains (i.e., physical exercise, medication adherence). Researchers have explored the design of ECAs that interact with users over multiple conversations, ranging from a handful of interactions to hundreds of interactions spanning months or years [11], [12] and [13]. Ring et al. [13] developed a conversational agent-based system to provide longitudinal social support to isolated older adults by means of empathic feedback. An exploratory short-term pilot study demonstrated significant reductions in loneliness of older adults based on self-reported affective state. Vardoulakis et al. investigated the use of an agent to provide social support and wellness counselling to older adults. Qualitative analysis of interactions with a remote-controlled agent (i.e., Wizard-of-Oz) installed in homes of older adults, identified multiple topics that users liked discussing and showed high acceptance ratings and a positive attitude towards the agent [14]. Other studies [15] and [16] explored relational agents, ECAs designed to form longterm social-emotional relationships with their users for health education and health behaviour change interventions. Results of a two month trial that investigated exercise promotion showed increased physical activity for participants using a virtual exercise coach compared to those using a conventional pedometer [17]. Nevertheless, this effect diminished when the coach was removed, suggesting that further research is needed to cause long-term behaviour change. In a similar study, Bickmore et al. developed a virtual laboratory to explore the longitudinal usage of a virtual exercise coach [18]. Older adult participants interacted with an agent from their home once a day for up to 120 days. Results showed that users who interacted with an agent that used variable dialogue exercised significantly more than those interacting with an ECA with non-variable dialogue. ECAs move beyond the paradigm of

computer as a tool and allow for multimodal interaction reflecting natural human-to-human communication. By exhibiting a certain level of intelligence and autonomy as well as social skills, ECAs provide familiar and non-threatening interfaces, especially useful for building systems that are easy to use, engaging and gain the trust of older adults.

From an architectural point of view it can be seen the ECA are realized in different approaches. The system can be designed as a single device. The device is in this case represented by the avatar.
According to an ambient intelligence approach it has been shown that the avatar or the embodied agent can have its major functionality in representing a distributed (in the environment) intelligence. Like shown it can be for users straighter forward to interact with a single point in a human like interaction. In this case the avatar device is connected to a whole network of intelligent devices providing calculation power as well as context information. The lot of information as well as processing power can in these cases even provided by a cloud. The advantages of such a system are:

- The system can grow with the needs of the person
- The system is flexible so software as well as hardware updates (as single nodes can be updated or added to the network)
- Functionality like avatar generation, emotion recognition, 3D object detection etc. can in this case be “outsourced” to devices in the network (or cloud) and have not be provided on the device which runs the avatar itself (e.g. tablet)
- The avatar can serve as a real representation of a distributed intelligence as the person can interact with the same avatar on different devices (tablet, mobile phone, tablet on the wall in the kitchen when cooking etc.)
- Services can be used by many clients as they are provided centralized (as web services). This also has several advantages regarding software maintenance and the potential for system extension

Due to the many advantages we choose for the Miraculous-Life system the distributed approach.

3.2 Service provision and reuse

Ambient Assisted Living (AAL) is emerging as technological support that can help elderly in their activities of daily living, enabling them to live more independently in their preferred environment. Examples of assistance can include giving reminders as a remedy to failing memory, facilitating social interaction with family and friends, monitoring doors, windows, water flow, and stove to improve security and safety in the home, tracking patients that wander off and detecting falls, and sharing of information between subject of care, healthcare personnel and next of kin. To provide the best assistance, these services need to be selected and personalized to each user, and furthermore this personalization may need updates as cognitive and physical condition tend to deteriorate over time. Even though the needs for assistance and care vary from person to person, there are similarities at a service level. This opens for reuse of a high percentage of the services, given that the services allows for personalization and flexible composition. In the next subsection, Service Oriented Architectures are presented as a technological solution that provides reuse and flexibility to the AAL domain [5].

Service Oriented Architectures (SOA) is an architectural software style that seeks to divide an information system into a set of interconnected stand-alone components that expose their functionality through a clearly defined interface. These components can be discovered and used by one or many applications through standard communication protocols such as the HTTP. The most common implementation of SOA is Web Services. Web Services are now being used by most organizations to share information across systems and platforms, using the Internet/Intranet as a transport medium. Gartner research estimates that Service-Oriented Architecture (SOA) "*will be used in more than 50 percent of new mission-critical operational applications and business processes designed in 2007 and in more than 80 percent by 2010*" [19].

The benefits of SOA are often reported based on key properties of the technology. Kawamoto and Lobach [20] highlights that SOA focuses on encapsulating core business capabilities within independent software services, and leveraging these services by using various business front-ends to fulfil business requirements. The SOA properties they list are; use of business oriented services, message-based interactions with "black-box" implementations; communication over a network; platform neutrality; service description and discovery; and loose coupling between system components. They argue that this provides the following benefits:

- Simpler software design and implementation, by decomposing complex problems into smaller, more manageable ones.
- Improves software reusability through enhanced reuse of existing IT resources.
- Improved adaptability to changing business requirements.
- Cost savings consequent to the above benefits.

Based on the outlined findings we have chosen the SOA approach for the services we provide in the Miraculous-Life System. All services are provided encapsulated and provided over a central server. The communication protocol is the described HTTP / WS protocol.

3.3 Smart Sensor Integration and Interoperability

For smart sensor integration the facets of interoperability, modularity and hardware independence are of uttermost importance. A platform with integrated sensor should be flexible and open for different smart home applications for different target groups and individuals based on a stable integration of sensor networks for home automation and medical devices [21].

Experiences in this field show that the avoidance of isolated or proprietary applications is an important aspect. Re-usability and easy customization of applications and infrastructure reduces costs in the elderly care domain. Standardization in communication between sensors and measurement devices is in the long-term a benefit for AAL developments. It enables interoperability and plug-and-play solutions. Furthermore data coming from devices by different manufacturers can be easily compared and one can combine different technological solutions. At the moment many manufacturers still use their own proprietary protocols and thus the integration is harder.

The mentioned modularity can be realized by using an OSGi framework in Java. Java based applications are platform independent. They are executed on the Java Runtime Environment (JRE), which can be installed on various operating systems. This fact is another aspect of hardware independence. The usage of an OSGi framework provides

remote maintenance and individual adaptability of the system. The components, coming in the form of bundles for deployment, can be remotely installed, started, stopped, updated and uninstalled without requiring a reboot of the system. Thus the framework is flexible in terms of expanding its functionality and updating single modules during runtime. The interactions and dependencies between bundles are handled by the framework itself. It manages searching and binding of required services, which are exposed functionalities within OSGi bundles, even when the service is activated at later time. Fine grained configuration options allow detailed access to functionalities in each OSGi bundle. The current OSGi specification version 4.3 is developed by the OSGi Alliance, which is an open standards organization [1]. Several OSGi implementations are available, e.g. Knopflerfish, Apache Felix, Equinox and Concierge OSGi.

Interoperability and compatibility of products are keywords in AAL technologies [21]. In the long-term the focus has to be on interoperability using off-the-shelf sensors and measurement devices by different manufacturers. In the field of AAL two standards have to be highlighted. On the one hand the ISO/IEEE 11073 standard representing a standard for health informatics in the field of health communication with the focus on establishing interoperability and on the other hand the ISO/IEC 14543-3. The latter also known as KNX technology is a worldwide standard for all applications in home and building control, ranging from lighting and shutter control to various security systems, heating, air conditioning, monitoring, alarming, water control and energy management [22]. Different associations (especial the Continua Health Alliance [23] for ISO/IEEE 11073 and the KNX Association for ISO/IEC 14543-3) make an effort to spread these standards.

The issues of modularity as well as sensor abstraction have been also addressed by the universaal project[1]. One outcome of universAAL is the open OSGi platform HOMER which has been provided as open source (see: homer.aaloa.org) by the Ambient Assisted Living Open Association, and has been further developed by the Miraculous-Life partner AIT. The HOMER platform is also used in Miraculous-Life as the platform for sensor abstraction and sensor integration. The usage of such platform middleware make Miraculous-Life also independent from the vendor of different sensors and at the same time open for a plug and play integration of new sensors and therefore the possibility of new use cases and services. A further explanation of the HOMER system in Miraculous-Life you can find in chapter 7 of this document.

The implementation of the described platform has shown that the usage of open standards and frameworks facilitates a simpler and more open software architecture for applications in the field of AAL. Several novel technologies are available in the field of software development and are already widely-used in the AAL domain and so as well in Miraculous-Life (e.g. OSGi, Maven, SpringDM). Ongoing developments especially in the open source community allow early integration of these technologies to facilitate important requirements like security, modularity and expandability. Furthermore one should emphasize the possibility for an adoption of the platform to use an open-source architecture, like the intended one by universAAL. This aspect will be especially supported by the modular architecture of the platform and the usage of technologies for software development as described above. Moreover the standardized interface for sensor integration is another reason not to be neglected. The aspect of standardization of the communication of sensors and measurement devices is in the long-term a benefit for the development and application of AAL technologies. It makes interoperability and plug-and-play solutions

[1] http://universaal.org/

possible. Furthermore data coming from devices by different manufacturers can be easily compared and one can combine different technological solutions.

4 Miraculous-Life System Requirements

4.1 Requirements for the Miraculous-Life system architecture

Several requirements regarding the technical solution are given from the DoW and the user requirements.

4.1.1 Requirements from the DoW

The system needs several **Input Devices** which are the physical part of the platform and gather inputs in real time from the user. The input devices should be integrated in a plug and play manner and the system should be flexible to have additional input devices integrated in future. The current Miraculous-Life system integrated several sensors installed in the environment of the user like infrared movement sensor, bed sensor, window contact sensors, a 3D depth sensors (Kinect), microphones and touch screens. These inputs solve also for the emotional multimodal interaction of the user with the VSP.

Additionally a **Real time multimodal processing** layer is needed to recognize and interpret the data input and process it and indicate the meaning to the intelligent modules in order to generate a coherent emotional response to the user.

The **Reasoning and Decision Making** layer consist of reasoning and decision taking modules. These rely on inputs from the previous layer, to determine what the system should communicate to the user whether through the avatar or the different services.

The **Application and Service Layer** contains a set of ICT services to provide the user services which he can use and a Social network to ensure the exchange of information with other but also with formal and informal caregivers.

The **Output Generator** is creating the Miraculous- Life VSP. It visualizes the VSP to the environment through the physical parts of system (3D graphics rendering and speech synthesizer).

The **Knowledge Base** of the system is serving as data management server where all data (e.g., information models, monitoring data, user profiles etc.) are stored and disseminated to cooperating services.

4.1.2 The Miraculous-Life system requirements following the user requirements

The user requirements regarding the system functionality of Miraculous-Life has been detailed described in deliverable D1.1. This deliverable also in detail described all technical and non-technical requirements for the system. The system requirements have been taken into account in the overall technical specification of the architecture. The following requirements have been found (for a more detailed description, please check deliverable D1.1):

Platform specification:

Table 3: Platform requirements and specification

Identifier	Requirement	How reflected in the ML architecture
R1	has to be designed to run on typical devices namely tablets and/or touch screen	Decentralized architecture, processing power outsourced. Tablets run only the UI component.

	computers	
R2	used on a wide range of user computers (with respect to processor speeds and graphics specifications)	The system is independent from the operating system as the UI is provided over a web browser
R3	user data to be kept in a secure location easy maintenance and system update	The user data are stored in a secure decentralized place (KB) applying several security measures (data channel encryption, user management etc.). The service functionalities are encapsulated in the OSGi container and can get started, replaced and enhanced independently.

Functional Requirements:

Table 4: Functional requirements

Identifier	Requirement	How reflected in the ML architecture
R4	Multimodal Input	System has been designed for several input devices: Kinect, microphones etc. It is important that the system is flexible regarding the input – the camera of the tablet can be used as well as from the Kinect – likewise for the microphone
R5	Multimodal Output	The system provides a HTML5 UI output displayed in the browser comprising visual output (avatar), an extended UI (text, buttons) as well as a voice output
R6	Facial Emotion Recognition	The system provides facial emotion recognition over the facial emotion recognition module. Both cameras (tablet as well as Kinect can be used)
R7	Behaviour Analysis	The system has integrated several layers of low and high level behavior analysis which are as well integrated as OSGi modules and can be updated and substituted in a straight forward way
R8	Real-Time Dialogue management	The system integrates a further developed real-time dialogue processing based on workflow implementations
R9	Real-Time Interaction	The user provides a real time interaction with the system over several input channels. The interaction is transmitted in real-time
R10	Graphical User Interface (GUI)	The graphical user interfaces is due to HTML5 very flexible regarding its design. It can easily be re-designed and adapted to different user needs. The UI is displayed over a browser. It's even thinkable that different UI designs are applied to the same ML system

		deployed. The UI design has been part of a user centered design and run through several design processes.
R11	Input sensors and data acquisition	The HOMER middleware applied in ML makes the system very flexible regarding the sensor integration of different kind of home automation sensors from different vendors. The HOMER platform supports also the ISO/IEEE 11073 standard.
R12	Continues use and maintenance	The modular and distributed approach of the architecture makes it possible to keep the system running also when different components are changed or updated. Especially the component deployment in the OSGi framework features this issue.
R13	Data logging	As all communication from the user with the system is managed by the dialogue manager he is logging several interactions of the user with the system. Additionally the Knowledge base which provides the service content is logging the usage of the ML services.
R14	Modularity and extensibility	The ML system follows a decentralized architecture following the SOA approach as well as deploying components inside a OSGi container.
R15	Dynamic Knowledge Base	The Knowledge Base is designed flexible and provides direct APIs for receiving data information. The Knowledge Base APIs are managed over a central service manager.

Non-Functional Requirements:

Table 5: Non-Functional requirements

Identifier	Requirement	How reflected in the ML architecture
R16	Privacy	Several privacy and security measures are applied in all layers of the system as well as the basic components (KB, protocols, buses etc.)
R17	Security	Several privacy and security measures are applied in all layers of the system as well as the basic components (KB, protocols, buses etc.)
R18	Performance	State of the art technology is used regarding hardware components as well as software tools. Performance is also achieved through the outsourcing of high performance components to external server.
R19	Reliability	The aim was to structure the architecture

		very clear. This means also that we used only minor very stable protocols for communication like AMQP, HTT, WS. Inside the OSGi container as own communication is maintained which is another advantage of the OSGi approach.
R20	Usability	Several usability measures have been taking into account during the whole project. We performed 3 pre-trials before the actual trials. The findings from the pre-trials have been taken into account in the re-design processes of the system
R21	Persistency	The aim was to structure the architecture very clear. This means also that we used only minor very stable protocols for communication like AMQP, HTTP, WS. Inside the OSGi container as own communication is maintained which is another advantage of the OSGi approach.

4.1 The Miraculous-Life Service-oriented architecture (SOA) approach

The goal of using SOA is to liberate the business from the constraints of technology without throwing the 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. An a SOA approach, applications are build using a set of building blocks known 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.

IBM 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.” [3],[4].

Many requirements described can be followed by a Service oriented architecture approach (SOA). The Miraculous-Life architecture tries to uncouple especially the service infrastructure to easily reuse service functionality or to extend it by new service functionality.

The Miraculous-Life SOA reference architecture is based on IBM’s reference architecture for Service-Oriented Architecture [3]. The reference architecture consists of five layers – each layer comprising a set of “component” that conforms to the rules and requirements specified for the layer. Furthermore, the service layer is grouped into five groups: application, domain and system specific group. This grouping is done to clearly separate the components that are specific for the applications (such as the use cases being implemented for Mirculous-Life).

The IBM Reference Architecture SOA is more in detail described in ANNEX A.

An overview of the Miraculous-Life SOA components can be seen in Figure 1.

According to the SOA paradigm, all hardware enterprise components (i.e., different sensor devices) compose of the bottom layer of the architecture. In order to interact with these devices, a Sensor Hardware Abstraction & API layer has been implemented. For this layer the open source OSGi framework HOMER has been re-used and adapted for Miraculous-Life. HOMER[2] has been developed as sensor abstraction framework in the universAAL[3] project.

HOMER has been selected because it allows integration of various sensor networks through appropriate drivers, which can be both open source and proprietary, depending on the field of application and availability. Its modules are distributed under the Apache license and can thus be utilized for business models of the business partners freely. In Miraculous-Life all home automation sensors have been implemented over this framework. This makes Miraculous-Life also open and independent from the vendor of window contact and bed sensors. The HOMER OSGi framework in that way provides a kind of Middleware layer abstracting the hardware to all the upper layers and services. The abstraction is based on the ISO/IEEE 11073 standard. This makes it possible that the sensors can be easily integrated via plug and play and that the overall system is independent from the sensor vendor as long as the used sensors follow this standard. Other sensors like microphones and Kinect are integrated directly with the SDKs provided by the vendor (Kinect) or the operating system hardware support (microphones/Android) etc.

The basic communication between the different modules in the upper layers is realized with the Noldus Communication Framework (NCF) which is based on the communication protocol AMQP.

The Knowledge Base runs independently and is providing all data as Java APIs. The data provided are (i.e. User profiles) and information models (i.e. Environment & Conversation Models).

Components which provide behaviour recognition and interpretation of the elderly monitoring are connected with the NCF framework.

The next layer presents all the services as web/OSGi services to the higher levels. All the services can use the components and infrastructure provided by the lower layers.

The high-level services or business services are service processes which use the service components for higher scenarios based on different single services.

The overall application and the VSP use different scenarios to provide the full functionality.

The security and privacy infrastructure is independent form the layers and integrated as general mechanisms or in the security relevant components itself. This includes enforcing identity and security in order to ensure that only authorized users may access and use sensitive data or invoke services. As common for platform services, security services are orthogonal to application and functional oriented services.

[2] http://homer.aaloa.org/
[3] http://universaal.org/index.php/en/

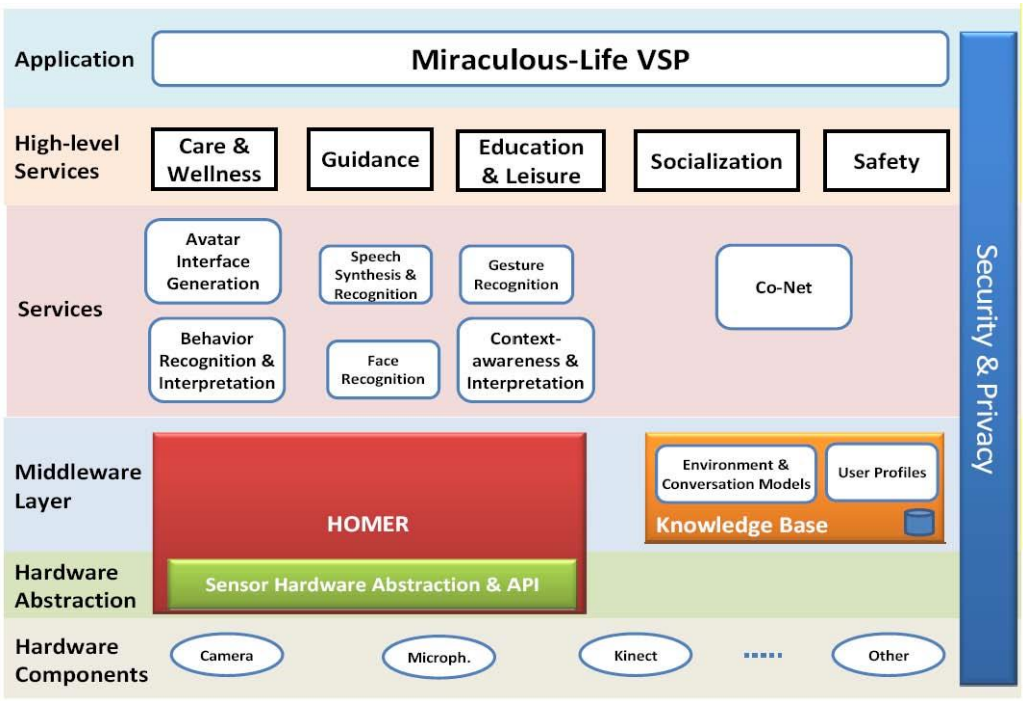


Figure 1: The Miraculous-Life module architecture following the SOA approach of IBM

5 Miraculous-Life Technological Overview

5.1 Introduction to this chapter

This chapter describes the technologies (hardware and software) employed in the Miraculous-Life project and provides a motivation for their use and an explanation on how they were used. Much care was taken to use technologies that are easy to use, reusable, and where possible are free of charge.

5.2 Hardware

5.2.1 Workstation

As workstation we refer to the computer on which the Miraculous-Life core system runs on. All interactions and communications, from the sensor input to the tablet output are processed on the workstation. The modules of the workstation are described in this deliverable. The setup of the workstation is described in deliverable D5.3. The specifications were chosen by incorporating different aspects like:

- User acceptance (unobtrusive, small, silent …)
- Current and possible future system requirements

The hardware requirements for using the Kinect for Windows v2 sensor are outlined in chapter 5.2.3. As it turns out, the Kinect sensor uses the most resources of the workstation, hence the given minimum system requirements had to be up scaled in order to provide sufficient processing power to fulfill the Miraculous-Life system requirements.

After extensive system testing, it was concluded to use a state of the art Intel Core i7 4790k CPU, along with 16 GB of RAM and an SSD Hard drive to ensure a fast and reliable system operation over an extended period of time.

Lower end CPU's were found to be inadequate, as the processor usage was at 100% as soon as all services that make use of the Kinect sensor were enabled. Letting a system run constantly at these very high loads, induces the risk of system failures, data loss and a reduction of the lifetime of the used hardware components.

More detailed hardware specifications can be found in the upcoming Deliverable 5.3c.

5.2.2 Server

A Server is used at the trial sites, which provides common functionalities that are used by all clients concurrently. This includes the Database for storing and handling user related data, a Webserver for hosting the caregiver as well as the elderly user interface, the AMQP Server implementation that enables the communication via the NCF and a server side oriented Java OSGi framework. The hardware requirements are focused on providing a fast and stable accessibility of the user data, network connectivity and availability for 24 hours per day.

The IT departments of the end user organizations provide a virtual server running within their local server cluster that can handle our requirements without much effort. Furthermore, automatic system backups are scheduled on a regular basis, to overcome any issues related to data loss.

5.2.3 Tablet

As main interaction device, handheld tablets where chosen. On these tablets, the Virtual Support Partner (VSP) is displayed next to detailed information from the provided

Services. The end-users will interact with the Miraculous-Life system by speech recognition and by direct touch input if needed.

The tablet specification used for the pre-trials can be found in Deliverable D5.3.

5.2.4 Sensors

Miraculous-Life intends to offer the end-users support by assisting in their everyday lives. In order for the system to interact with the user, it needs to get information from the user to determine the current status and the next actions that have to be taken. Therefore, different types of sensors are needed which collect the necessary data in a mostly passive way. Great care must be taken at this part in order to avoid too much intrusion and/or interference with the end-users daily life activities and their privacy. The elderly must not feel observed or monitored by the system, which would decrease the acceptance of the system and therefore undermines the benefit of using it. During the project, no new sensors will be developed; instead already available, established and used sensors will be used to accomplish the project goals.

Microsoft Kinect v2

Microsoft Kinect is a sensor input device, which provides an innovative way of interacting with a game console or computer via gesture and spoken commands. Microsoft provides an SDK for using the raw data collected by the Kinect for further gesture and skeleton analysis. Miraculous-Life will use version 2 of the Kinect, which has superior hardware integrated compared to version 1.

To avoid blowing up this document with technical specifications of Kinect that is available online, please visit the official website [22].

Microphone

The built-in microphone array of the Kinect as well as the Tablet microphone will be used for audio data capturing. Additionally, if the audio quality is not good enough, an additional high quality microphone may be used via an USB connector to the workstation. Miraculous-Life will use the audio data as user input for speech commands as well as deriving the users' emotional state from it. The Microsoft speech recognition engine will be used for the speech recognition part while Zoobe provides the emotion analysis at their own server infrastructure.

Contact Sensors

Contact sensors will be connected through the HOMER platform to the Miraculous-Life System. The gathered data will be used as additional input for low level safety services.

Bed/Pressure Sensor

A bed or pressure sensor will be used to get valuable information about the sleeping and resting behavior of the elderly. This sensor is placed under the mattress and reacts on applied pressure when a person is lying down. This information is further on used by various services as additional input parameter.

5.2.5 External Servers

Zoobe Server

One of the project partners, Zoobe, is using their own Server for specific tasks, which are the emotion recognition from audio data as well as the Avatar generation.

Asterisk telephone Server

In order to provide calling functionality, we are using VoIP technology handled an Asterisk telephone Server. VoIP calls are usually much cheaper than calling via a regular phone line, while it is still possible to call regular numbers. Furthermore, VoIP can be integrated nicely in the User Interface of ICT based system, such as Miraculous-Life.

Fortunately, the hardware requirements for running a telephone server are very low. The main aspect that needs to be covered is the network bandwidth available for handling multiple calls at the same time. Having too less bandwidth available can significantly reduce the quality of the calls, by hearing chopping sounds or even experience call drops.

For further information about the Asterisk telephone Server requirements, please consult the upcoming Deliverable D5.3c

5.3 Software

5.3.1 Operating Systems

Workstation

As operating system, Miraculous-Life workstation will use Windows 7 Professional 32 or 64Bit edition. As the new Kinect 2 required a Windows 8 operating system it is forseen that we switch to that system after the pre-trials have been completed. Since the complete system requirements (also regarding the hardware) are not yet known (image / video processing, skeleton data analysis etc.) it is unclear if there is a need for more than 4GB of RAM and therefore a 64Bit operating system. For the pre-trials, a 32Bit System is sufficient, since the chosen scenarios don't require complex computations.

Windows as itself is needed due to some software components provided by the project partners, which are only runnable on computers that run a Windows operating system, such as the Microsoft Kinect Speech Recognition engine or the Microsoft SQL Server.

Server

The Server is running on Microsoft Windows Server 2012 R2, a server operating system that is supposed to run for an extended period of time without interruption. As Database system, the relational database management system Microsoft SQL Server 2012 (Developer Edition) is used to store all user and other essential data.The KnowledgeBase component, which is developed by UCY, enables the insertion, selection, update and deletion of database entries to the other system components via a specified API. For further information about the KnowledgeBase, please refer to the corresponding Deliverable.

Tablet

The tablets operate on Android 4.4. For the final prototype, Android 5 should be available on the tablets, which brings a lot of useful and desirable features compared to the older Android version. For example, the Android 4.4 webview which is used to present the elderly user interface does provide full support to all features of HTML5. For now, a 3rd party framework is used to enable all relevant HTML5 features to properly use the user interface.

In Android 5, the native webview is running on the Google Chrome Browser engine, which provides a lot of features from the new HTML5 standard, like the WebSocket or the webRTC protocols.

5.3.2 Java OSGi

The Java driven software components in the Miraculous-Life system will be development by complying with the OSGi specification, release 4.3. This approach allows for a highly dynamic and flexible development of software modules so that new functionalities, enhancements and modifications are easy to integrate.

Modularity is realized by using an OSGi framework in Java. Java based applications are platform independent. They are executed on the Java Runtime Environment (JRE), which can be installed on various operating systems. This fact is another aspect of hardware independence. The usage of an OSGi framework provides remote maintenance and individual adaptability of the system. The components, coming in the form of bundles for deployment, can be remotely installed, started, stopped, updated and uninstalled without requiring rebooting the system. Thus, the framework is flexible in terms of expanding its functionality and updating single modules during runtime. The interactions and dependencies between bundles are handled by the framework itself. It manages searching and binding of required services, which are exposed functionalities within OSGi bundles, even when the service is activated at later time. Fine grained configuration options allow detailed access to functionalities in each OSGi bundle.

Apache Karaf

The Apache Karaf Framework[4] sits on top of the OSGi container. Karaf adds additional functionality to OSGi via add-on features. You can control the container directly through direct shell, SSH or a web console. It supports provisioning and deployment via file system, Maven repository, archive, OBR (OSGi Bundle Repository) and configuration via file system, web console and JMX. Dependency injection using Blueprint is also available in Apache Karaf. Additionally, you can deploy bundles and features at runtime very easily by putting them in the “%HOMERframework%\deploy\” folder.

For further information, please have a look at:

- Apache Karaf Manual: http://karaf.apache.org/manual/latest-2.3.x
- Learning Apache Karaf [2]

Apache Karaf is a small OSGi based runtime which provides a lightweight container onto which various components and applications can be deployed.

This list shows some features supported by the Karaf:

- **Hot deployment:** Karaf supports hot deployment of OSGi bundles by monitoring jar files inside the [home]/deploy directory. Each time a jar is copied in this folder, it will be installed inside the runtime. You can then update or delete it and changes will be handled automatically. In addition, the Karaf also supports exploded bundles and custom deployers (blueprint and spring ones are included by default).
- **Dynamic configuration:** Services are usually configured through the ConfigurationAdmin OSGi service. Such configuration can be defined in Karaf using

[4] http://karaf.apache.org

property files inside the [home]/etc directory. These configurations are monitored and changes on the properties files will be propagated to the services.

- **Logging System:** using a centralized logging back end supported by Log4J, Karaf supports a number of different APIs (JDK 1.4, JCL, SLF4J, Avalon, Tomcat, OSGi)
- **Provisioning:** Provisioning of libraries or applications can be done through a number of different ways, by which they will be downloaded locally, installed and started.
- **Native OS integration:** Karaf can be integrated into your own Operating System as a service so that the lifecycle will be bound to your Operating System.
- **Extensible Shell console:** Karaf features a nice text console where you can manage the services, install new applications or libraries and manage their state. This shell is easily extensible by deploying new commands dynamically along with new features or applications.
- **Remote access:** use any SSH client to connect to Karaf and issue commands in the console
- **Security framework based on JAAS**
- **Managing instances:** Karaf provides simple commands for managing multiple instances. You can easily create, delete, start and stop instances of Karaf through the console.

Apache Maven

Apache Maven (http://maven.apache.org) is a software project management and comprehension tool. Based on the concept of a project object model (POM) Maven can manage the compilation of source code, the project's building process, reporting and documentation from a central configuration file. The maven-eclipse plug-in facilitates creation of Eclipse projects to import source code packages in Eclipse easily.

Java projects can be automatically built in a way to meet OSGi requirements due to the maven-bundle plug-in.

Furthermore, Maven supports dependency management by downloading external artefacts from private or public repositories on demand during the build process and can also publish the generated applications on remote sites.

5.3.3 WebSocket

The WebSocket protocol[5] enables a full-duplex communication channel over a single TCP connection. The communication is initiated by the client and established via an HTTP handshake with an appropriate update request. When the update request is acknowledged by the webserver, the bi-directional communication channel can be used until one of the partners closes the connection. In contrast to a HTTP connection, the server can send a message to a client without being polled by the client beforehand, which is used in the Miraculous-Life project to update the tablets' user interface.

A main advantage of using the WebSocket protocol instead of the HTTP is the much lower overhead throughout the entire connection, due to the omission of the handshake

[5] http://tools.ietf.org/html/rfc6455

procedure every time the client wants to talk to the server. The whole communication becomes more fluid and responsive by using this new and state of the art protocol.

5.3.4 JSON

JavaScript Object Notion (JSON)[6] is a data format used to transmit data objects consisting of „key“:“value” pairs, which are used to exchange messages over the NCF as well as for the WebSocket connections. Example message structures will be shown in the appropriate Deliverable sections where the data exchange is further explained.

5.3.5 HTML5

The HyperText Markup Language v5 (HTML5)[7] is used for displaying data in a browser. The Miraculous-Life System uses HTML5 primarily on the tablets, since the newest version of the HTML standard has certain functionalities that are highly beneficial for achieving the project goals. For example, HTML5 introduces a WebSocket protocol implementation as well as an already integrated video tag for displaying videos without the need of additional browser plugins. This enables the development of a highly flexible, responsive and dynamic user interface, with the additional advantage reusing the same interface on different devices that have HTML5 support.

5.3.6 CSS

Cascading Style Sheets (CSS)[8] are used to describe the look and format of a markup language, like HTML. In Miraculous-Life, CSS3 (Level 3) will be used to design the User Interface (UI) on the tablet.

5.3.7 JavaScript

To enable a dynamic interaction for the otherwise static (HTML presented) UI, JavaSript (JS)[9] is used. JS is a scripting language that enables web browsers to execute client-side scripts which give users dynamic interaction possibilities with the UI. JS provides the functionality to initiate a WebSocket connection, therefore acting as a WebSocket client, which is used to present new information to the end-user without the need of reloading the whole UI but only updating distinct parts of it.

Furthermore, with the introduction and implementation of the webRTC (web real time communication[10]) protocol, it is now possible to use VoIP directly in a browser, without the need for a 3rd party software (usually called a softphone). As this communication path is handled through Javascript, it is possible to embed the calling functionality directly within the user interface, which is a huge improvement to previous approaches via 3rd party software.

Finally, as an outlook for future developments, the webRTC protocol also enables the possibility of video data transfer, making it possible to have a live video chat with another party from within the user interface and again, without the need of additional software.

[6] http://json.org/
[7] http://www.w3.org/TR/html5/
[8] http://www.w3.org/TR/CSS/
[9] http://www.ecma-international.org/publications/standards/Ecma-262.htm
[10] http://w3c.github.io/webrtc-pc/

6 Miraculous-Life architecture

6.1 Introduction to this chapter

This chapter explains the overall concrete architecture of the Miraculous-Life setup. The chapter explains all the modules and the functionalities as well as the overall structure.

6.2 Miraculous-Life concrete architecture

The following section shows the Miraculous-Life modules and the communication protocols as well as physical deployment. In general the Miraculous-Life components are distributed in internal and external devices. The internal devices and modules are deployed and installed inside the elderly's flat or living compartment. The internal deployed modules and devices are visualized in Figure 2. The external modules and components are deployed externally in a central workstation inside the residential house or even outside the internal network as external services. The external deployed modules and devices are visualized in Figure 3.

In the elderly home the setup consists of a small PC, a tablet as well as several sensors (Kinect, microphone, home automation sensors). The PC (workstation, OS windows 7 – later versions windows 8) runs the user recognition modules as well as an OSGi Apache Karaf environment to connect the sensors in a smart and standardized way (ISO/IEEE 11073). Moreover, the speech recognition module, the human behaviour analysis as well as the environment context analysis is running on the PC as well. The PC also hosts the VSP Multimodal Dialogue Management Framework as well as a utility module (both running in the OSGi environment). The tablet handles the displays of the avatar (VSP) as well as manual button interaction or text presentation if needed. This information is presented as an HTML5 webpage to the user.

The inner module communication is based on the NCF (Noldus communication framework), which uses the AMQP implementation RabbitMQ as its core component. The tablet is connected to the workstation by a wireless TCP/IP connection.

The external devices and modules combine the following. The server provided by the project partner Zoobe. This server is responsible for the Avatar generation as well as the emotion recognition based on the audio signal. A central Miraculous-Life server is providing all the services (deployed in OSGi Karaf) as well as a central database. This central server can be placed for example aa residential care service facility and serve many clients (living units of the older adults). This provides the advantage of a central service management (updates, etc.) as well as a central database for all information (also connected information of the end-users and their relatives in the residential care organization). As all the services are require a client ID, the central services can serve multiple clients.

Table 6: Information to the architecture figuresTable 6 shows some information for the following architecture and module concepts.

Table 6: Information to the architecture figures

Blue dotted lines	Specifies distinct devices
<...>	This notation indicates the communication protocol used by the connection
<px>	Communication protocol is not defined yet
<TCP/IP>	Communication protocol will be based on TCP/IP but is not yet further defined
Black connections	Represent connections from Sensors to capture components
Green connections	WP2 outputs; Communication to other components is established by using the **NCF** or, if the components are written in Java, via OSGi EventAdmin Service or the ServiceRegistry. If the output goes directly to the KnowledgeBase, a database connector like ODBC or JDBC can be used
Red connections	WP3 outputs;
Light Blue connections	WP4 outputs; Services will be OSGi based services and will utilize the OSGi Service registry for providing their service
Magenta connections	related WP unknown; Communication used for the User Interface
Orange connections	Communication used by the Co-Net

General information	• The speech recognition will be done using Microsoft speech recognition engine.
	• The emotion recognition from audio will be done on Zoobe Servers
	• Zoobe uses their own Server to render and generate the Avatar and its video file, respectively.
	• System conclusion with the two previously mentioned developments:
	  ○ An Internet connection is **mandatory** for the system:
	  ▪ to display an (not default) Avatar that is interacting with the end user
	  ▪ to have a working emotion recognition from audio
	• UCY will provide an API for the Knowledge Base
	• The KnowledgeBase will be located on a server (located at the user-organizations) later on in project, but is locally for the pre-trials
	• The "Display" Element is not related to a specific WP yet. It is used to display and interact with the end user.
	• The "Utility" Element provides general modules, like e.g. a WebSocket Server (which is included in the HOMER runtime) and a local video storage
	• The Service column "General Services" includes the three

	defined more generic services: ○ Reminder Service ○ Alert Service ○ Notification Service ○ as well as other more general services like ▪ UpdateWebUI • The more specific reminder, alert and notification services are therefore utilizing the "General Services" • Elements within the distinct "Apache Karaf OSGi environment" Blocks can communicate with each other via the ○ OSGi EventAdmin Service (also in a distributed environment (DOSGi)) ○ OSGi ServiceRegistry (also in a distributed environment (DOSGi))

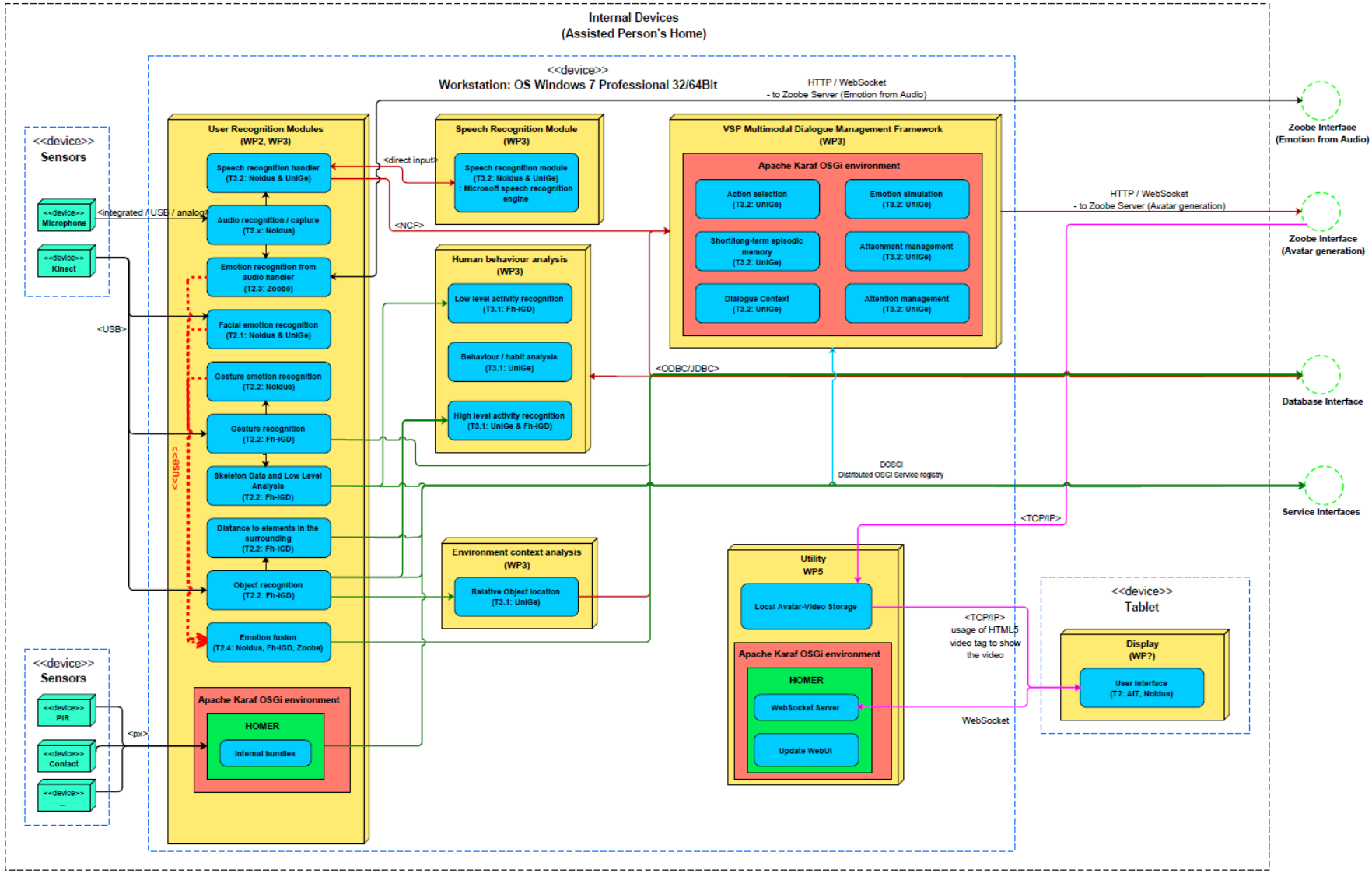


Figure 2: The Miraculous-Life modules running internal in the elderly person home / flat environment

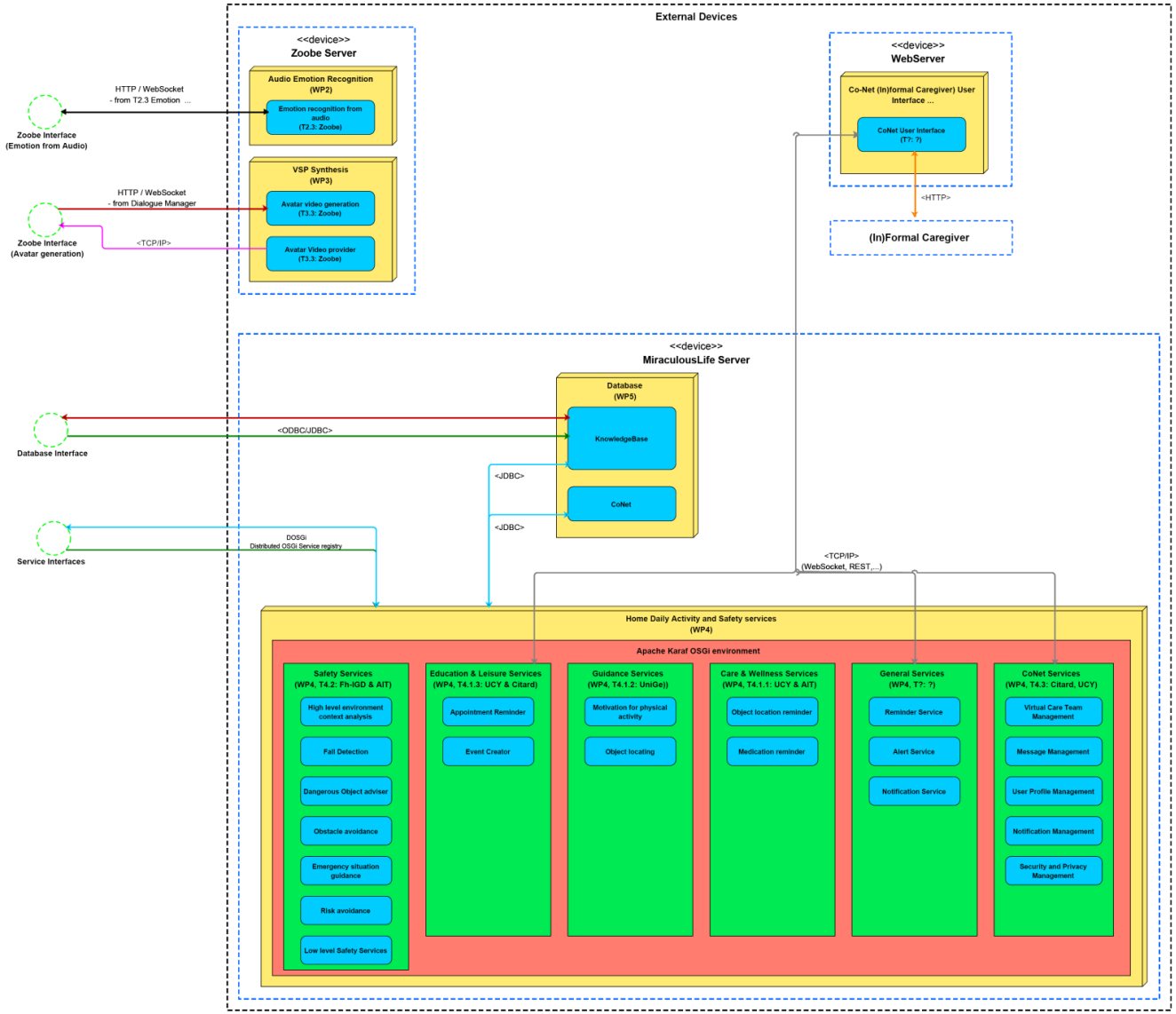


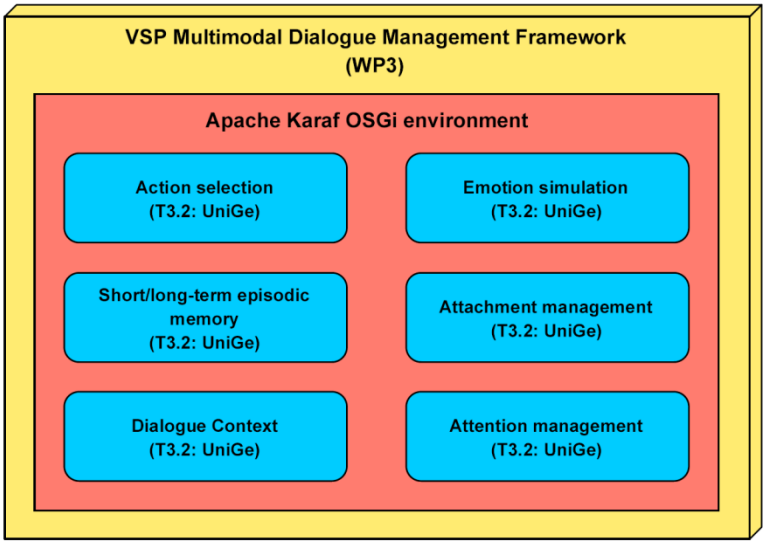
Figure 3: The Miraculous-Life modules running external which means outside the elderly person home / flat environment

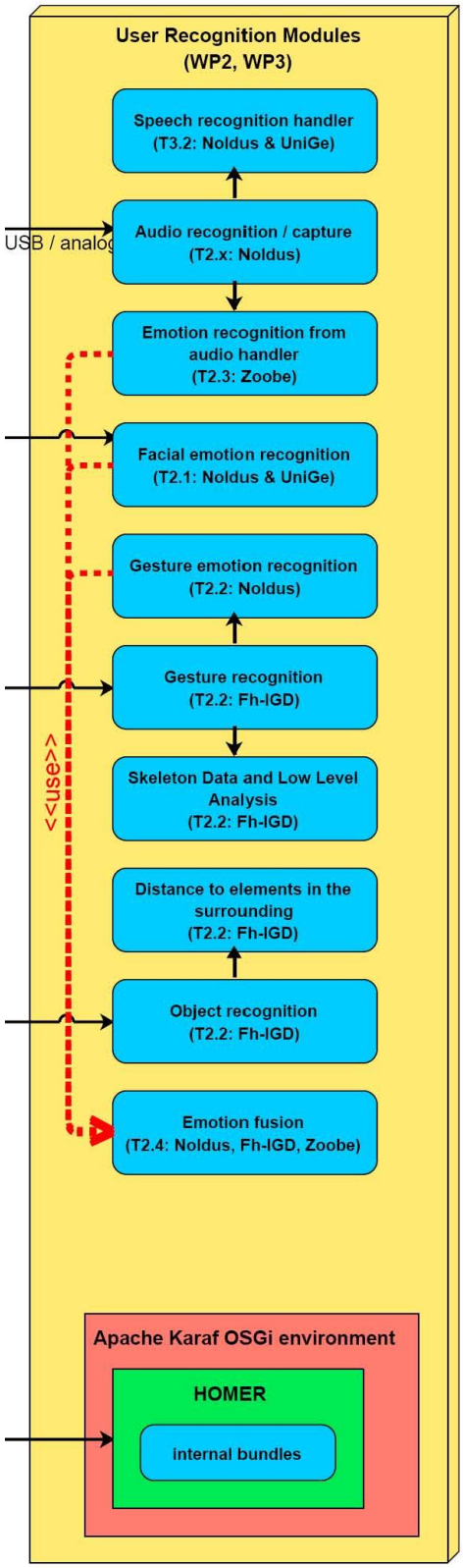
6.2.1 The internal setup modules

The **user recognition modules** are developed by the work packages WP2 and WP3. They retrieve the input from the sensors such as camera, microphone or the Kinect sensor. This module block contains a lot of modules which perform a first recognition of persons and objects as well as gestures, facial expression and emotions. Normally the modules are connected to the external sensors (Kinect, camera and microphone) by USB.

This module block also contains the HOMER[11] (home recognition bundles) components, which are running on the OSGi environment. The HOMER bundles are modules developed as an open source project by AIT and allow a vendor independent integration of home automation sensors (abstraction to the ISO/IEEE 11073 standard) as well as a pre-processing of behaviors of the user

The **VSP Multimodal Dialogue Management Framework** is the artificial intelligence behind the avatar. Its builds up the dialogue management as well as the decision making. It comprises also the modules, which simulates an emotional avatar. This module block is connected via the NCF (Noldus communication framework) to other modules and



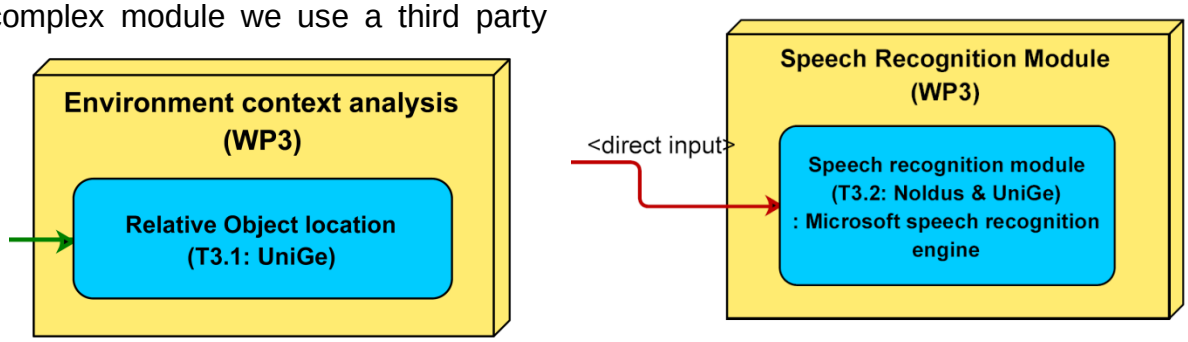


sends the requests for the Avatar generation directly to the Zoobe server, which generates the Avatar. Further, this module integrates the services into the interaction scenario by implementing a Java reflector

[11] http://homer.aaloa.org/

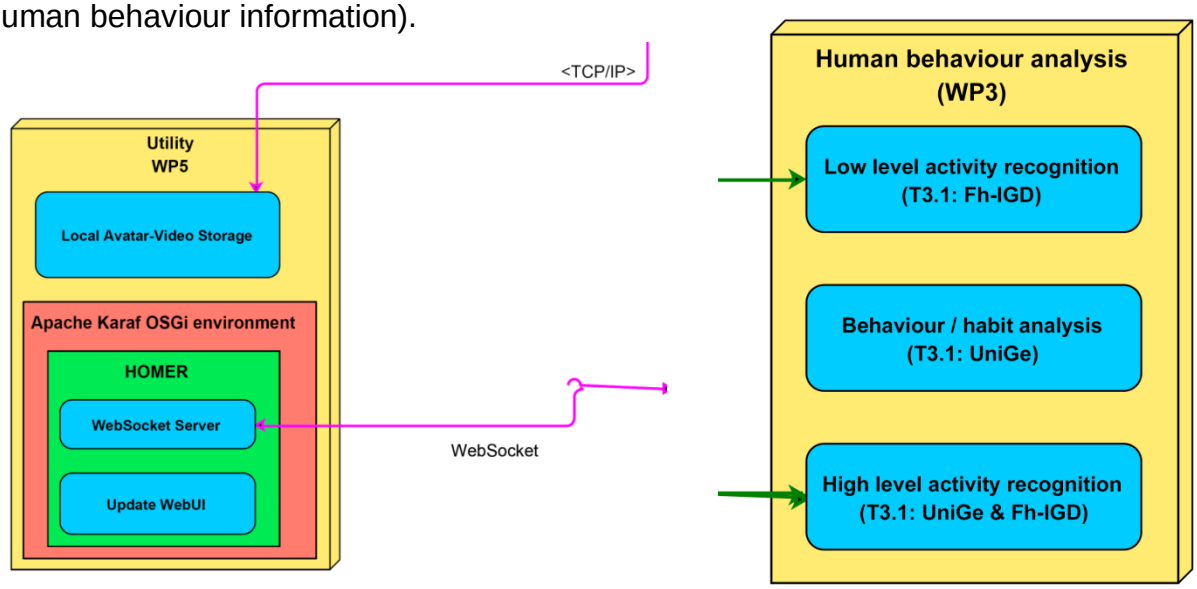
mechanism than enables dialogue scripts to dynamically load services and execute their methods.

The **Speech Recognition Module** generates the speech input from the user and interprets the input signal. For this complex module we use a third party



engine from Microsoft. The speech recognition module is connected via the NCF with the speech recognition handler in the User Recognition Modules.

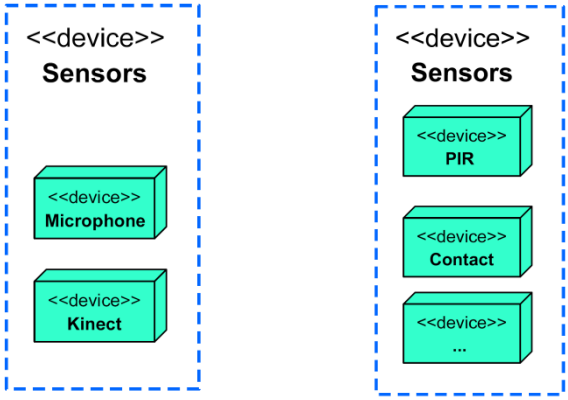
Based on low level object location information, coming from the object recognition module in the User Recognition Modules, the **Environment context analysis** module gives concrete information about the object and the place (high level information). The **Human behaviour analysis** module takes the low level information from the User Recognition Module (skeleton data, object information etc.) to at the one hand abstract user behaviour as well as high level activity (logical information combination of object information as well as human behaviour information).



The **Utility Module** is coordinating the display to the end user (tablet). It loads the Avatar video generated by the external Zoobe server, which generates the Avatar, and generates an HTML5 page combined with the graphical UI (displaying things like Agenda, input buttons etc. on demand). The combined display is visualized on the tablet or multiple tablets depending on the demand.

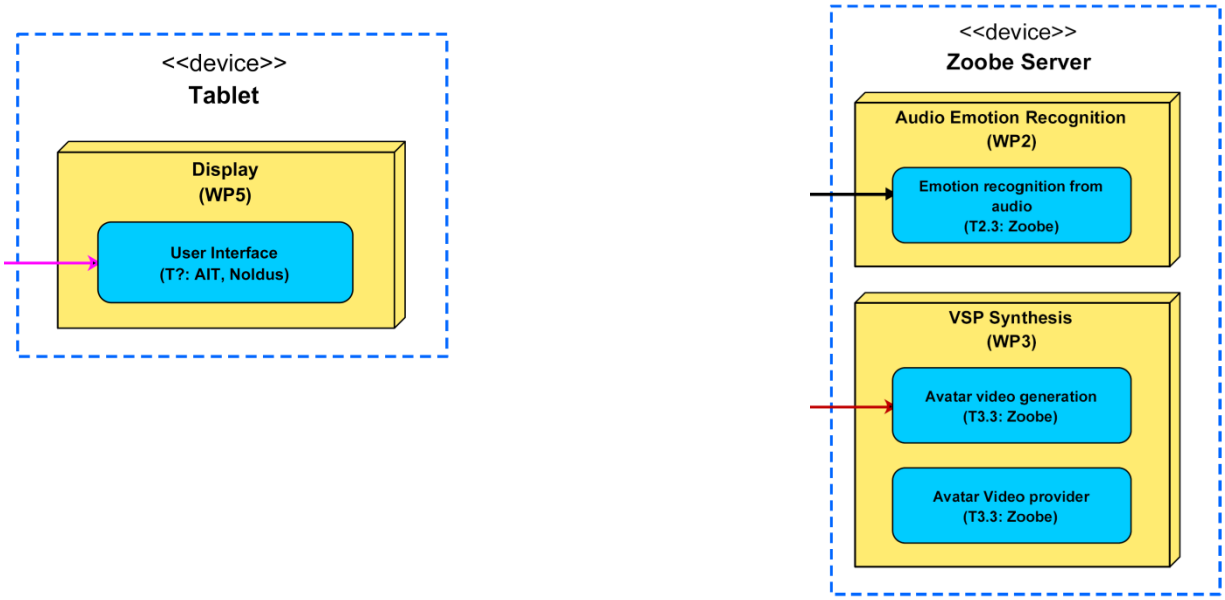
The user interface for the elderly is a **tablet**. As all the UI interface is provided by a web socket the system can easily be extended by multiple tablets. For later versions of Miraculous-Life, it is planned to have a second (or more) bigger touch screen mounted on the wall in the kitchen or the living room (or any other place). With this setup, the Avatar would be more present in the environment of the elderly and would come closer to a more omnipresent virtual living companion.

Sensors are connected directly to the respective driver modules in the User recogniton modules or via the HOMER OSGi modules.



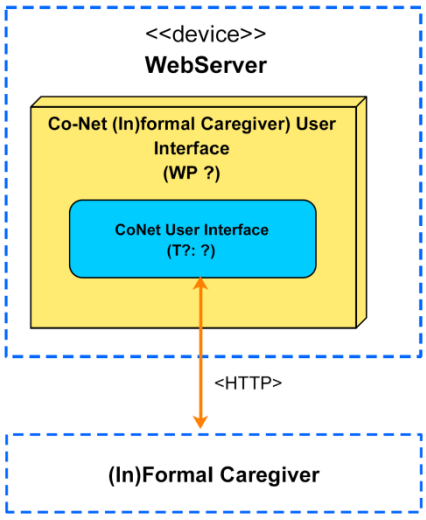
6.2.2 The external setup components

External (not in the living units of the elderly) installed modules are basically three components. The central Miraculous-Life server is for the project setups running on a central place in the internal network of the end user residential complexes. This server serves all the Miraculous-Life clients inside the elderly living compartment and host all the general services. An additional web server outside the internal network is running the Co-Net services and the database. This functionality will be in later iterations be integrated in the central server. A third server, provided by the project partner Zoobe, is performing the Avatar generation as well as the emotion recognition based on speech input.



The **Zoobe Server** is directly connected, via HTTP / Web socket, to the emotion recognition from audio handler in the User Recognition Modules. The VSP synthesis is connected via HTTP / Web socket to the VSP modules as well as the Utility module (TCP /IP) which uses the rendered Avatar and does embed it in the UI for the end user.

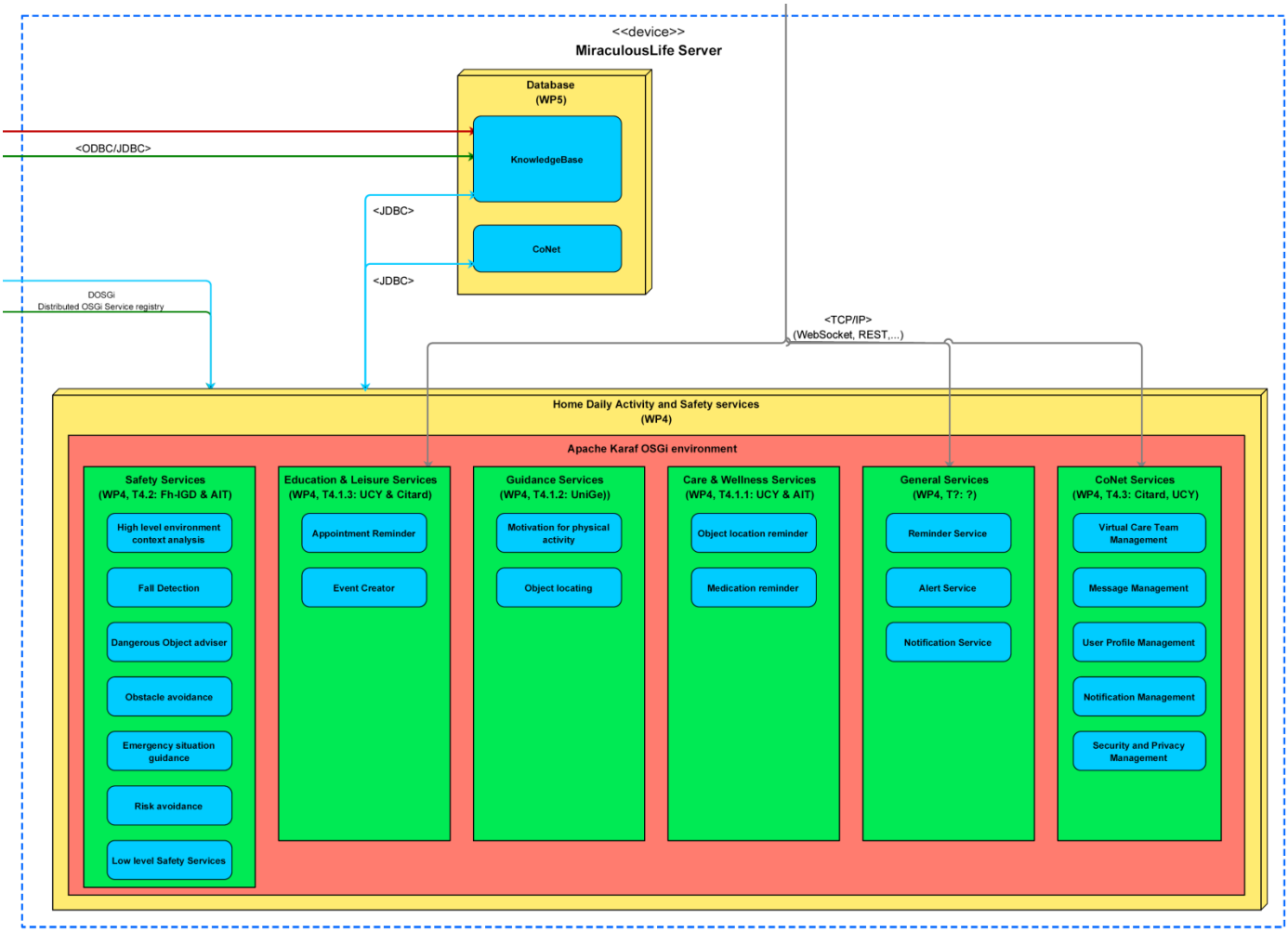
The **Co-Net** access is provided by an external web server. The Co-Net is providing all information to the formal and informal caregivers about the elderly. The Co-Net is also providing and running the web services which the caregivers can use to insert information about the elderly and the elderly network. We plan to separate the interfaces for the elderly from the interfaces for the caregivers as the caregivers will not interact with an Avatar and have different preferences regarding the look and feel as well as the demand on input as well as output information.



The **Miraculous- Life server** is running the central database as well as all services in the different service categories:

- Safety Service
- Education & Leisure Services
- Guidance Services
- Care & Wellness Services
- General Services

All services will be deployed on top of the Apache Karaf OSGi environment and will serve all Miraculous-Life clients in the different elderly compartments (connection is TCP/IP websockets).



7 HOMe Event Recognition System (HOMER)

7.1 Introduction to this chapter

The following chapter explains the HOMER OSGi framework. This framework is an open source project and can be used to integrate sensors and devices from different vendors and with different communication protocols via an international standard. This technology allows the setup of and home platform already provides basic rules for pattern recognition.

7.2 Description of HOMER

The HOMe Event Recognition System (HOMER) integrates the local (off-the-shelf) sensors and performs pre-processing. This open source platform is based on an Apache Karaf OSGi framework and encapsulates its functionalities in terms of OSGi bundles, which enables modularity. The bundles are executed on the Java Runtime Environment (JRE), which can be installed on various operating systems, what offers hardware independency. The usage of an OSGi framework provides remote maintenance and individual adaptability of the system. The components, coming in the form of bundles for deployment, can be remotely installed, started, stopped, updated and uninstalled without requiring a reboot of the system. Thus, the framework is flexible in terms of expanding its functionality and updating single modules during runtime. The interactions and dependencies between bundles are handled by the framework itself. It manages searching and binding of required services, which are exposed functionalities within OSGi bundles, even when the service is activated at later time. Fine grained configuration options allow detailed access to
functionalities in each OSGi bundle. Along with OSGi several supporting technologies, like Apache Maven[12] and Eclipse Gemini Blueprint[13] (formerly known as Spring Dynamic Modules[14], which was migrated to the Eclipse foundation) are used. Standards for medical device communication and home automation networks are integrated to enable communication to appropriate devices. All of these technologies are used to realize important aspects for an AAL service platform, namely security, modularity, extendibility and interoperability. Furthermore HOMER makes use of several standards,
namely:

- Independent Living Activity Hub specialization ISO/IEEE 11073-10471.
- ISO/IEC 14543-3: KNX is a standardized OSI-based network communications protocol for intelligent buildings[15] .

Applied to all connected sensor technologies is a mapping within HOMER components to one central, standardized data model. This is essential for further data processing in terms of event recognition and reasoning. Here are a few examples of scenarios you can use HOMER for:

- Changing status of devices (on/off)
- Energy consumption monitoring
- Warnings on open doors or running devices when leaving home
- Calendar and reminders
- Person tracking

[12] http://maven.apache.org/
[13] http://www.eclipse.org/gemini/blueprint/
[14] http://docs.spring.io/spring-osgi/docs/current/reference/html/
[15] http://en.wikipedia.org/wiki/KNX_(standard)

- Activity index
- ...and many more!

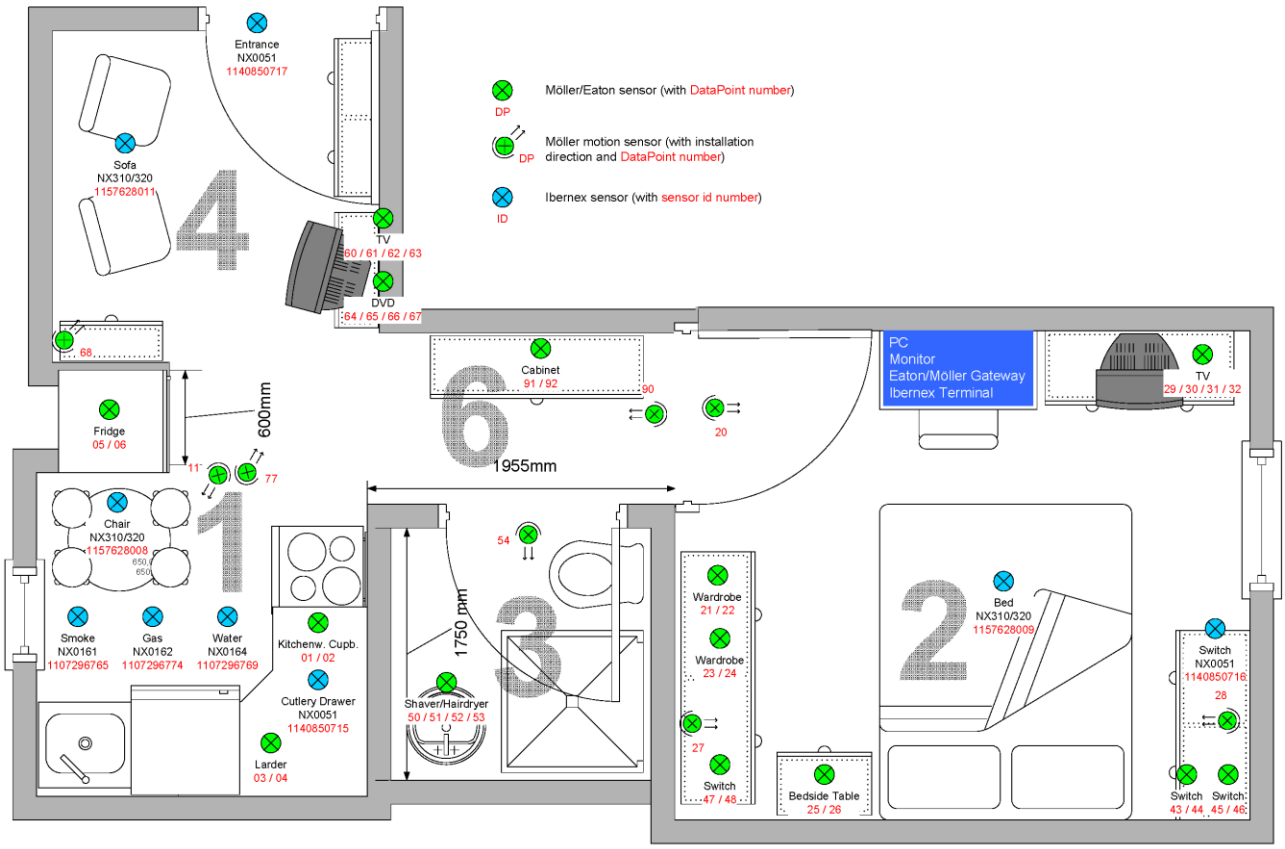


Figure 4: Example flat used with sensor locations

7.3 License

HOMER Core is an Open Source project, licensed under the GNU LESSER GENERAL PUBLIC LICENSE V2.1. The Homer OSGi framework can only be used in compliance with the License. A copy of the license can be obtained at http://www.gnu.org/licenses/lgpl-2.1.txt. Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an “AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

7.4 HOMER architecture

HOMER is developed in Java and follows the OSGi (Open Services Gateway initiative) specification [1]. Also HOMER is based on Apache Karaf. The advantage of OSGi is the modularization of the whole application into a number of smaller bundles, which can provide services to each other. You actually can start/stop/update/exchange modules during runtime.

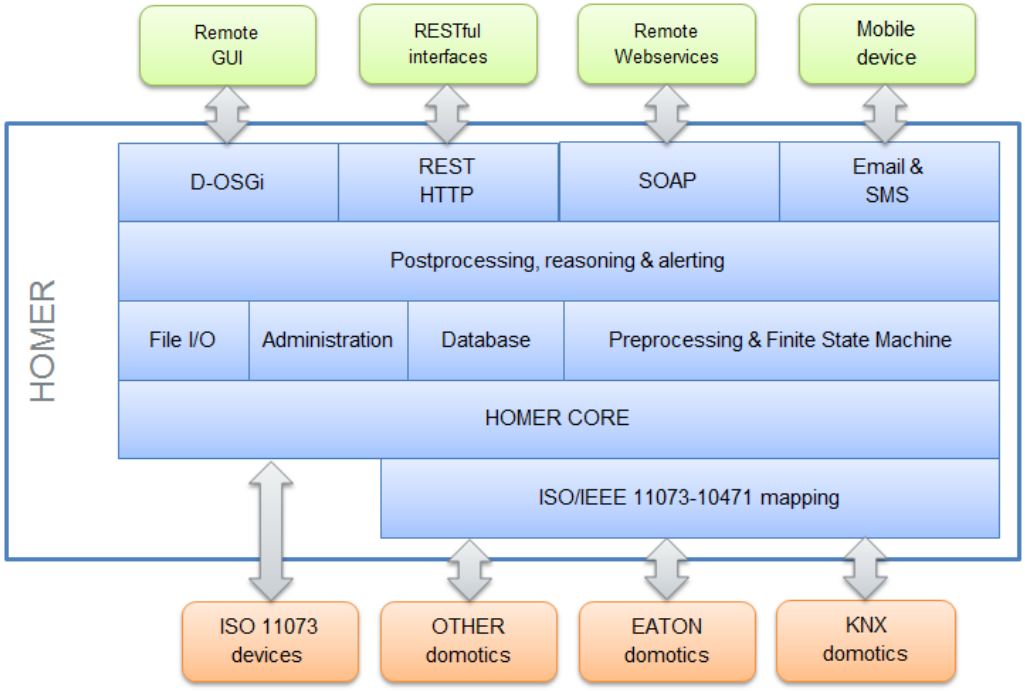


Figure 5: HOMER system architecture

The HOMER binary distribution is able to run on any platform that supports Java SE and was tested on the following platforms:

- Windows:
  - Windows 8
  - Windows 7
  - Windows Vista
  - Windows XP SP2
- Unix:
  - Ubuntu Linux
  - Debian Linux

In order to run HOMER the Java SE Development Kit (Version 7) has to be installed. Furthermore the JAVA_HOME environment variable must be set to the directory where the Java runtime is installed.

8 NCF

8.1 Introduction

This chapter explains what the Noldus Communication Framework (NCF) is, gives an overview of the components used and shows how it can be used.

8.2 Description

The Noldus Communication Framework (NCF) consists of two parts; A service to handle communication and a client library (available in C/C++, C# and Java) that implements the communication functionality. The idea behind the NCF is to deliver a platform and language independent way of communication between multiple processes (running either on the same machine or on multiple machines).

8.3 Architecture

NCF uses a RabbitMQ service (https://www.rabbitmq.com/), which is a message broker, to host the communication between NCF clients. The server uses AMQP (Advanced Message Queuing Protocol), which is a wire level protocol and is built on top of TCP/IP, to communicate with other AMQP enabled applications. AMQP is an open standard for passing messages between applications. It is implemented by RabbitMQ, which is the distributed message broker of choice for the NCF, since it is open source, fast, scalable, highly configurable (e.g. guarantee message delivery) and platform independent. For a client to be able to communicate with other NCF enabled clients, it has to use the NCF client library. The library includes the functionality to send and receive data. The client library implements the NCF protocol on top of AMQP.

8.4 Usage scenarios

NCF supports two different types of messaging:

- **Publish-Subscribe**. In this type of message pattern there is a publisher that creates messages and there are zero, one or more listeners that subscribe to those messages. The publisher is not aware of whom is listening. Note that it can be set up in such a way that only one listener will receive the messages.
- **Request-Reply.** Publish-Subscribe is a one way messaging approach. In a Request-Reply scenario. A publisher creates a request and waits for a response on this request. On the other side of there is a listener that receives this request and will reply with a response.

9 Miraculous-Life services

Miraculous-Life deploys a number of ICT and Safety services in order to support the daily home activities of the elderly. As illustrated in Figure 1, Miraculous-Life currently supports a number of services that can be roughly classified in the following categories:

- **Care & Wellness Services**, whose objective is to provide support and care to the elderly. Some representative examples of services in this category include: a) *agenda* based on a simplified realization of the RFC 5545 specification); b) *medication management* service; and c) *household adviser*.
- **Guidance Services**, whose objective is to provide for stimulation and support to the elderly in order to maintain a healthy lifestyle. Some representative examples of services in this category include: a) motivation for *physical exercises*; and b) *assistance in locating objects* at home.
- **Education & Leisure Services**, whose objective is to educate the elder and also provide recreational activities. Some representative examples of services in this category include: a) meal preparation; and b) events and group activities.
- **Safety Services**, whose objective is to detect possibly dangerous situations using high level environment context analysis and to facilitate emergency workflows to aid the elderly. Some representative examples of services in this category include: a) *fall detection* using the kinect camera; b) *obstacle avoidance*; and c) *reminders* for turning off potentially dangerous devices (e.g., cooking stove).

Additionally, Miraculous-Life utilizes the Collaborate Care Network tool (Co-Net), which promotes collaboration between the elder and formal/informal carers Co-Net maintains a unique personalized profile for each elder, allows the creation and management of Virtual Care Teams (VCTs) around the elderly and facilitates communication between the elderly, and between the elderly and caregivers.

Finally, Miraculous-Life employs a manager component in order to provide easy access to all services.

All Miraculous-Life services are described in D4.1 in detail.

10 Overview of Miraculous-Life architecture components and artefacts

Table 7: Components and artifacts

ML component	Module category	Achievement in ML
Audio recognition / capture	User recognition module	Developed in ML
Emotion recognition from audio handler	User recognition module	Developed in ML
Facial emotion recognition	User recognition module	Further developed in ML based on a Noldus component
Gesture emotion recognition	User recognition module	Further developed in ML based on a Noldus component
Gesture recognition	User recognition module	Further developed in ML based on a Fh component
Skeleton Data and Low Level Analysis	User recognition module	Developed in ML
Distance to elements in the surrounding	User recognition module	Developed in ML
Object recognition	User recognition module	Developed in ML
Emotion fusion	User recognition module	Developed in ML
Action Selection	VSP Multimodal Dialogue Management Framework	Further developed in ML based on a UniGe component
Emotion simulation	VSP Multimodal Dialogue Management Framework	Further developed in ML based on a component from UniGe
Short / Long term episodic memory	VSP Multimodal Dialogue Management Framework	Further developed in ML based on a component from UniGe
Attachment Management	VSP Multimodal Dialogue Management Framework	Developed in ML
Dialogue Context	VSP Multimodal Dialogue Management Framework	Developed in ML
Attention management	VSP Multimodal Dialogue Management Framework	Further developed in ML based on a component from UniGe
Relative Object Location	Environment context analysis	Developed in ML
Speech recognition module	Speech recognition Module	Implemented in ML using the Microsoft speech recognition engine
Low Level activity recognition	Human behavior analysis	Developed in ML
Behavior / habit analysis	Human behavior analysis	Developed in ML

High level activity recognition	Human behavior analysis	Developed in ML
Low Level Avatar Storage	Utility	Developed in ML
HOMER websocket server	Utility	Further developed based on a open source component from AIT
Web UI	Utility	Developed in ML
Environmental Sensor integration	Sensor integration	Further developed based on a open source component from AIT
Emotion recognition from Audio	External Devices / Zoobe Server	Developed in ML
Avatar video generation	External Devices / Zoobe Server	Further developed based on a commercial component from Zoobe
Avatar video provider	External Devices / Zoobe Server	Further developed based on a commercial component from Zoobe
Co-Net services	Co-Net server	Further developed based on a component from Citard
High level environment context analysis	Miraculous-Life services	Developed in ML
Fall Detection	Miraculous-Life services	Developed in ML
Dangerous Object Adviser	Miraculous-Life services	Developed in ML
Obstacle Avoidance	Miraculous-Life services	Developed in ML
Emergency situation guidance	Miraculous-Life services	Developed in ML
Risk avoidance	Miraculous-Life services	Developed in ML
Low Level safety Service	Miraculous-Life services	Developed in ML
Appointment Reminder	Miraculous-Life services	Developed in ML
Event Creator	Miraculous-Life services	Developed in ML
Motivation for physical Activity	Miraculous-Life services	Developed in ML
Object locating	Miraculous-Life services	Developed in ML
Object location reminder	Miraculous-Life services	Developed in ML
Medication reminder	Miraculous-Life services	Developed in ML
Reminder service	Miraculous-Life services	Developed in ML
Alert service	Miraculous-Life services	Developed in ML
Notification service	Miraculous-Life services	Developed in ML
Virtual Care Team Management	Miraculous-Life services	Developed in ML
Message Management	Miraculous-Life services	Developed in ML
User Profile Management	Miraculous-Life services	Developed in ML
Notification Management	Miraculous-Life services	Developed in ML
Security and Privacy Management	Miraculous-Life services	Developed in ML
Knowledge Base	Database	Developed in ML
CoNet services	CoNet	Further developed in ML

		based on components from Citard
HOMER	Infrastructure	Further developed and adapted for ML based on an open source sensor integration framework developed by AIT
Noldus Communication Framework	Infrastructure	Noldus communication protocol which has been re-used as communication protocol in ML

11 Security and privacy infrastructure and considerations

The following table gives a brief overview of the security functionalities in Miraculous-Life.

Table 8: Security infrastructure in ML

Service	Description
Access Control	Access Control includes the Authentication and Authorisation services. The Authentication service verifies a user's credentials and allows access to the system only to users with valid credentials. The Authorisation service determines what operations and which data an authenticated user can access, allowing access to resources only to legitimate, authorised users. An Authentication with the system is provided over the caregiver UI which have access to the system data
Encryption	The encryption mechanism describes functionality related to confidentiality and integrity protection of data. Data channel encryption is applied in the KB.
Secure Communication	Secure communication is the mechanism describing functionality needed to secure data being transmitted between two endpoints.

In Miraculous-Life the OSGi technology has been selected as the main technical platform for the implementation of services and components. Within the OSGi framework the SOA Services technologies are the main technological platform to use as presented in this Miraculous-Life architecture deliverable.

11.1 Related Security Standard

SSL/TLS

SSL [24] and TLS [25] are point-to-point protocols used to provide confidentiality and integrity protection of data between two communicating entities.

Miraculous-Life can benefit from SSL/TLS if secure communication is to be achieved between two entities with direct communication.

11.2 Security in Miraculous-Life

11.2.1 Security Requirements

The following table gives a brief overview of the security requirements in Miraculous-Life.

Table 9: Security requirements

Requirement	Type	Comments
Guarantee information confidentiality	Encryption/Authentication/Authorization	Information stored in local servers. Accessed only by authorized personnel (authentication).

Isolate system from external access	Privacy	Internal servers should not be accessed from Internet. A firewall will be used to manage incoming and outgoing connection on specific ports.
Data analysis and exploitation	Privacy	Data should not contain personal information and should be treated statistically. Only information suitable for processing may be exploited and analysed.
User personalization	Privacy	Allow user to decide at all times if he wants to be recorded or not. Provide the means to start and stop being captured at any time.
Secure communications	Privacy/Encryption	Communications could be encrypted with secure protocols (SSL, TLS) if necessary.
Good practices	Privacy	IT departments in charge of securitization should follow the usual good practices when managing Miraculous-Life servers and DB.

11.2.2 Stakeholder Roles

We consider the following user profiles are recommended for the Miraculous-Life system:

Table 10: User types

User type	Description
Regular user	Individual users that have been authenticated by the system
System administrator	System administrators with modification permissions, for instance to erase a particular user profile when a user requests to do so.

It is necessary to classify different types of information in categories to protect user's privacy. These categories are based in the degree of confidentiality of information. The following table summarizes the needed categories, showing the categories with the highest confidentiality level on top:

Table 11: Data categories

Data category	Content	Accessible by
Confidential	Passwords and encrypted information	Individual users
Private	Personal information such as user name and personal data (introduced at register process)	Individual users and system administrators
Shared	Info that can be shared among users	Individual users and system administrators that have access permission.

Public	Anonymous information that may be used for statistical analysis	It can be accessed by all users of the system

Considering these categories, all the information in the database must be classified to make them fall in one of them and treat them accordingly.

For each of the categories there will be a different level of security required:

- Confidential data will always be encrypted and only accessible to individual users to whom the information belongs.
- Private information needs not to be encrypted but it will only be visible to individual authenticated users or to systems admins who want to delete personal information associated to a user who wants to do so. In this way, shared and public information from the user could be preserved without compromising his confidentiality, as this info will no longer be associated to a particular user.
- Shared information does not need either to be encrypted and it will be visible to authenticated users of the system with whom the information has been shared.
- Public information will be available to all users and it may be used for data analysis and exploitation of the information.

Information will be stored only in local servers, protected from external access by running a firewall.

11.2.3 Platform layers and security

As was discussed earlier in this deliverable, the SOA system design approach facilitates distinct layers that operate and interact with each other. At the same time, the Security and Privacy layer is considered to be orthogonal to the SOA layers, hence performs its functionality concurrently on all system layers (see also Figure 1: The Miraculous-Life module architecture following the SOA approach of IBM).

Hardware components

The data gathered by the different sensors like the Kinect, Tablet (both including microphone), contact and bed sensor are handled anonymously throughout the system. The sensor data received by the workstation is associating to a user only by a PersonID that is provided by the KnowledgeBase. That way, without having access to the database, the data cannot be linked to e.g. a person name.

Furthermore, the recorded audio data as well as the video data is not stored in the system. Instead, live data is processed immediately as soon as it is provided to the system and the outcome of this data processing, like specific artifacts or environmental information are stored in the database for later use.

Middleware Layer

The same principle of anonymous data handling as was explained for the hardware components is also applied in this layer. Additionally, every HOMER instance has its own auto generated ID that identifies each instance within multiple Miraculous-Life clients. The HOMER instance is associated with a specific user at the database level.

A separate security mechanism is additionally in place, which is the security framework within the Java Virtual Machine (JVM) itself and an extension provided by the OSGi framework. This security measurement is a general aspect of every Java based program

and has therefore no specific impact on the overall security and privacy aspect of the Miraculous-Life system. For this reason, it is not further explained in this document, but for the enthusiastic reader we want to refer to the official OSGi specification for further details [24].

Alongside the HOMER framework, the Database also resides in this SOA Layer. Only authorized users have access to the Database, and only a subset of those has permissions to actually view the contents of the database tables. The system users (elderly and caregiver) don't have direct access to the database, but can only interact and make use of it via the services and strictly defined API's. Furthermore, it was agreed on not encrypting the whole database, as this is quite resources intensive, but instead encrypting the data channel when querying for specific data sets. This ensures that the raw and sensitive data is transferred on a secure channel and cannot be sniffed by 3rd parties.

Services

The same principle of anonymous data handling as was explained for the hardware components is also applied in this layer. Furthermore, some services are using external services on remote servers that require a network connection, like the emotion recognition from audio data by Zoobe. For security and privacy reasons, these connection are established by using SSL (secured sockets layer) technology via the WebSocket protocol. This ensures that the data sent is encrypted and cannot be read by 3rd parties but only by the desired receiving terminal.

Some services make use of the raw audio and video data of the users, which is again not stored or permanently saved in order to maintain user privacy.

High-level Services

The higher level Services make use of already derived data from sensor input or from existing data in the database, both of which are already anonymized by that time. Based on their functionality, no further security and privacy measurements have to be taken, since these services are only dealing of ID's that can only be associated with a real user name in the Database.

Application

The Application layer provides on the one hand the VSP that is interaction with the elderly, and on the other hand a web interface for the caregivers. The caregiver interface is handled by the Co-Net component, which is described in more detail in Section 11.3.

Every hardware device in the Miraculous-Life system has a unique ID assigned, which can be associated with a specific user through the caregiver interface. The relation between a device and a user is only visible directly in the database or in the caregiver interface, both of which require explicit permission to gain access.

The elderly UI is provided as an HTML5 webpage, which connects via a secured (SSL) WebSocket (wss) connection to the Server. This ensures an encrypted transfer of data from the workstation to the user interface.

11.3 Co-Net privacy and security component

Personal data, related only to the socialization and profile aspect of the elderly, is the core resource that the Co-Net's services are built on. For this reason, security and privacy is

important for Co-Net, as it is crucial for the users to feel that the system does not allow unwanted privacy intrusions and to ensure that their sensitive data privacy is respected.

The most prominent aspect is that only authorized users may access and use sensitive data of the users. Based on this concept, Co-Net allows access to data never directly but only through a strict API. Additionally, only members of the user's Virtual Care Teams with specific roles, responsibilities and access rights can have predefined limited access to the user's data. Thus, the Co-Net's privacy and security component enables the security and privacy of user's sensitive data (managed by Co-Net, e.g., data related to the social and profile aspect of the users) by ensuring that only authorized users (i.e., users that are already authenticated by the Miraculous-Life platform) with certain access rights may access and use sensitive data of the users.

In deliverable D1.3 "Ethical, Privacy, Legal Considerations and Deontological practice", the security and privacy requirements relevant for the Miraculous-Life system have been presented. Based on these requirements, identification, authentication and authorization of users, as well as the integrity and privacy of the users' sensitive data should be respected.

To address the functional requirements mentioned above, Miraculous-Life system supports functionality for authentication and authorization using the Secure Sockets Layer (SSL) protocol. SSL is a standard security technology for establishing an encrypted link between a server and a client, typically a web server (website) and a browser, ensuring that any information communicated through this link can be understood only by the intended recipient. Thus, by using the SSL protocol, we create a secure connection between a user's web browser and the server of the Miraculous-Life system so as to prevent any unwanted privacy intrusion on the users' sensitive data, such as names, addresses, phone numbers, medication, calendar, messages and login credentials, and ensure that their data privacy is respected.

Authorization in the Miraculous-life system is based on tokens. Based on this concept, when a user logs in to the Miraculous-Life system, will first be authenticated (by providing a username and a password), by using the Authentication service (this is implemented and performed by the Miraculous-Life platform). Upon successful login, the system will issue a security token and associate it with the particular user (i.e., the caregiver or the avatar of elderly). This token will then be used to ensure that only authorized users (i.e., users that were already authenticated by the Miraculous-Life platform) with certain access rights will have access to invoke services/methods that use or manipulate sensitive data of the users. Every time a user (e.g., the caregivers or the avatar of the elderly) wants to invoke one of Co-Net's services/methods, the token will be validated and the user will be authorized to use the service/method. If the user's token is valid, then the authorization service will determine if the user has the specific role that will provide him/her the access right to use the specific service/method that he/she is trying to invoke. When a Co-Net's service/method is invoked, the Authorization service will validate if the user has the rights to invoke the specific service/method. That is, to validate if the user has the access rights to invoke a service/method, the specific service/method will call the authorization service to check if the token provided by the user is valid. If yes, authorization service will check the specific access rights that are associated with the specific user's role, and decide if the user is allowed to execute the service/method that is attempting to invoke. When access to the system services is no longer needed, the user can logout from the system. By performing a logout, the user's token will be removed from the system database, thus forcing the user to re-enter his/her credentials when access to the system services is needed in a later time.

A more detailed description of the Co-Net's privacy and security component as well as the database and the methods that this component supports is provided in deliverable D4.2b ("Specification of Co-Net").

12 Conclusion

The virtual support partner of the Miraculous-Life project provides a promising approach of a virtual conversational agent which can support elderly people in their familiar environment and provide them with services which enhance their quality of life. More importantly this approach evaluates how elderly people accept the conversation with an artificial intelligence and are willing to use services provided by the avatar. First evaluations showed promising results. Although a dialogue between an avatar and a human being cannot replace a dialogue between two human beings, this kind of solution might be more natural way to interact with artificial systems. This of course not only applies for the setting in elderly support but also in any interaction with ICT systems. The described system setup has been designed to provide a lightweight as well as adaptable system. Due to the system design it is possible to use the VSP on a single, standalone device independent from the operation system (tablets etc.). The services and database components which might need regular updates and extensions are centralized on a server which is serving many clients. This provides the possibility to update as well to extend the system (by services, clients etc.) in an easy way. The technology chosen is state of the art and easy to use. The service deployment in OSGi supports the idea of an easy software life cycle management.

The Miraculous-Life system follows a straight forward architecture with highly encapsulated components. This brings the advantage of easily extending the system; integrate new components or updating components regarding software (software management) as well as hardware.
In this way the Miraculous-Life system can also grow i) in its functionality, which is important in case the elderlies have other needs with growing older and ii) regarding his hardware, in case new and better hardware developments appear on the market.
The system can be extended with additional nodes of sensors and even other nodes not planed in the system so far (robots etc.).
Functionalities with heavy processing power like the avatar generation or the emotion recognition have been outsources to more powerful devices. At the same time the user is experiencing the avatar on one (or more) tablets – devices with a rather limited calculation power so far.
In this way the avatar serves as a real embodied agent of an intelligent environment which gets it context information and capabilities out of processing devices and sensors distributed in the environment.
We further conclude that SOA in development of support systems for elderly and their carers (formal and informal) can be beneficial when the services are designed in line with the domain needs and technical environment factors such as network stability are considered carefully. Software companies and system integrators could benefit from both our reported experiences and the reusable software services. The services running on a centralized server can serve in this way many clients in one residential house.

Beside this positive aspects it is clear that this advantages make the system very depending on the inter components communication. This means in the first place the wireless network as well as the internal chosen communication bus (NCF). Especially in case components are distributed in different virtual environments as well as physical environment this leads to several communication problems. This drawback should be considered when following the suggested architecture.

We are convinced that the chosen system provides a good possibility to further investigate the research question on Avatar interactions. The system provides many functionalities and components which also in future can be used and updated to further investigate in Avatar interaction. As described, different components can be easily updated in case of other technology or higher developed components. This has been at the end a major aim of the Miraculous-Life system.

References

[1] OSGi Alliance. *OSGi Service Platform Core Specification Release 4.3*. 2011. URL:

http://www.osgi.org/Download/Release4V43

[2] Johan Edstrom, Jamie Goodyear, and Heath Kesler. *Learning Apache Karaf*. Packt

Publishing Ltd, 2013

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

[4] Stale Walderhaug, Erlend Stav, Marius Mikalsen, Sten Hanke, *D1.1 Mpower Overall Architecture*, 2008

[5] E Stav, S Walderhaug, M Mikalsen, S Hanke, I Benc Development and evaluation of SOA-based AAL services in real-life environments: A case study and lessons learned - International journal of medical informatics, 2013

[6] Cassell, J.: More Than Just Another Pretty Face: Embodied Conversational Interface Agents. Commun. ACM, vol. 43, pp. 70-78, (2000)

[7] Wilks, Y.: Close Engagements with Arti_cial Companions Key social, psychological, ethical and design issues. John Benjamins Pub. Co., Amsterdam/Philadelphia, (2010)

[8] A. Ortiz, P. Carretero, D. Oyarzun, J.J. Yanguas, C. Buiza, M.F. Gonzalez, and I. Etxeberria, Elderly Users in Ambient Intelligence: Does an Avatar Improve the Interaction?. Intelligence, vol. 4397, pp. 99-114, 2002.

[9] A. Nijholt, Disappearing computers, social actors and embodied agents. in Proceedings of the International Conference on Cyberworlds, pp. 128-134, 2003.

[10] T.W. Bickmore, D. Schulman and L. Yin, Maintaining Engagement in Long-term Interventions with Relational Agents. Appl. Artif. Intell., vol. 24(6), pp. 648-666, 2010.

[11] T.W. Bickmore, Relational Agents: Effecting Change through Human- Computer Relationships. Massachusetts Institute of Technology, 2003.

[12] H. Pinto, Y. Wilks, R. Catizone and A. Dingli, The senior companion multiagent dialogue system. In Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems, vol. 3, pp. 1245-1248, 2008.

[13] L. Ring, B. Barry, K. Totzke and T. Bickmore, Addressing Loneliness and Isolation in Older Adults. In Affective Computing and Intelligent Interaction, 2013.

[14 L.P. Vardoulakis, L. Ring, B. Barry, C.L. Sidner and T. Bickmore, Designing relational agents as long term social companions for older adults. In Proceedings of the 12th international conference on Intelligent Virtual Agents, 2012, vol. 7502, pp. 289-302, 2012.

[15] T.W. Bickmore, A. Gruber and R.W. Picard, Establishing the computerpatient working alliance in automated health behavior change interventions. Patient Educ. Couns., vol. 59(1), pp. 21-30, 2005.

[16] T.W. Bickmore and R.W. Picard, Establishing and Maintaining Long-Term Human-Computer Relationships. ACM Trans. Comput. Hum. Interact., vol. 12, pp. 293-327, 2005.

[17] T.W. Bickmore, R.A. Silliman, K. Nelson, D.M. Cheng, M. Winter, L. Henault and M.K. Paasche-Orlow, A randomized controlled trial of an automated exercise coach for older adults. J. Am. Geriatr. Soc., vol. 61(10), pp. 1676-83, 2013.

[18] T.W. Bickmore and D. Schulman, A Virtual Laboratory for Studying Long-term Relationships between Humans and Virtual Agents. In Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems, pp. 297-304, 2009.

[19] C. Pettey, Gartner Says SOA Will Be Used in More Than 50 1583 Percent of New Mission-Critical Operational Applications and Business Processes Designed in 2007, Garther Media Relations 2007 Press Releases, 2007.

[20] K. Kawamoto, D. Lobach, Proposal for fulfilling strategic objectives of the US roadmap for national action on decision support through a service-oriented architecture leveraging HL7 services, Journal of the American Medical Informatics Association 14 (2007) 146–155.

[21] T Fuxreiter, C Mayer, S Hanke, M Gira, M Sili, J Kropf . A modular platform for event recognition in smart homes - e-Health Networking Applications and Services (Healthcom), 2010 12th IEEE International Conference 2010

[22] http://www.microsoft.com/en-us/kinectforwindows/meetkinect/features.aspx

[23] KNX Association, KNX Standard - Introduction. http://www.knx.org/ [Accessed: December 2015].

[23] Continua Health Alliance, Continua - Introduction. http://www. continuaalliance.org/ [Accessed: December 2015].

[24] "The SSL Protocol Version 3.0," 1996, [online]: http://wp.netscape.com/eng/ssl3/draft302.txt

[25] "The TLS Protocol Version 1.0," 1999, [online]: http://www.ietf.org/rfc/rfc2246.txt

ANNEX A

A.1 An architectural template for SOA [3]

The IBM Reference architecture defines 7 layers. They are presented on the next picture and description of the each layer is given below.

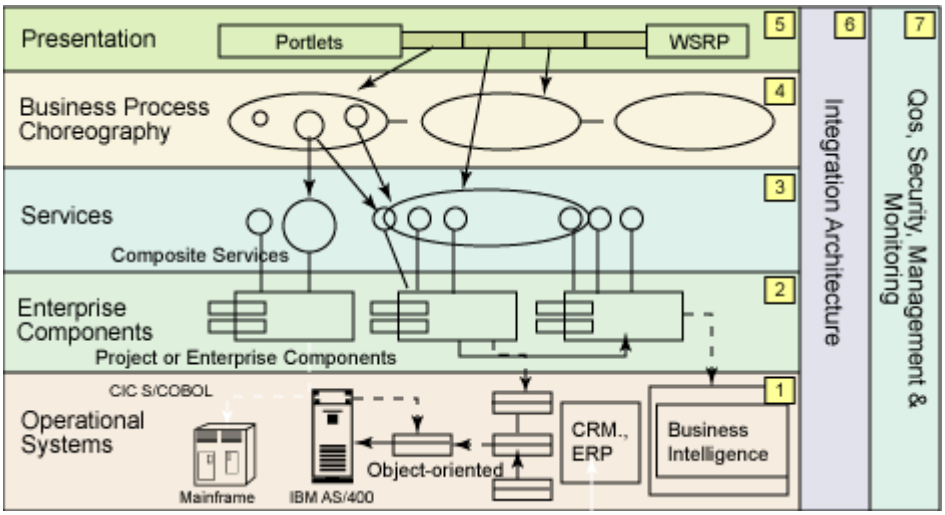


Figure 6: IBM SOA Reference Architecture

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 an 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). 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. Web Services Description Language (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. 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.