



<p>Contract no. 034707 www.mpower-project.eu</p>  <h1 style="margin-left: 200px;">m·power</h1> <p style="text-align: center;">Middleware Platform for eMPOWERing cognitive disabled and elderly</p>	 <p>SIXTH FRAMEWORK PROGRAMME: PRIORITY 2.5.11 eINCLUSION SPECIFIC TARGETED RESEARCH OR INNOVATION PROJECT</p>
<div style="background-color: black; color: white; padding: 5px; display: inline-block;">MPOWER Project Deliverable:</div> <div style="color: red; font-size: 1.2em; font-weight: bold;">Overall Architecture</div>	<p>Deliverable id:</p> <div style="color: red; font-size: 1.5em; font-weight: bold; text-align: center;">D1.1</div>

Key Information from "Description of Work" (from the Contract)

Deliverable Description	This deliverable describes the structure and underlying concepts for the MPOWER overall architecture. The description is focused around the Service-Oriented Reference Architecture, the MPOWER services and actors, and the information models that are used. A short description on how to apply the architecture in terms of model driven development is provided as background.
Dissemination Level	CO=Confidential (Consortium Members + Commission)
Deliverable Type	R = Report
Original due date (month number/date)	Month 6/15/21
Release number/date	V0.1 2007.04.01

Authorship Information

Editor (person/ partner):	Ståle Walderhaug / SINTEF Erlend Stav / SINTEF
Partners contributing	ETK, ARC, UCY
Reviewed by (person/ partner)	

Checked by and released (person/partner)	Marius Mikalsen /SINTEF
Date of release	2008-10-14

Release History

Release number	Date issued	Milestone	Approved by name/org	Document type: either "Master" or Annex Id	eRoom version	Release description /changes made
0.1	April 1, 2007			Master	14	First phase deliverable of D1.1
0.2	September 26, 2007			Master	19	Second deliverable of D1.1. Inserted HL7 dynamic SOA diagrams and deployment platform details
0.3	October 14, 2008			Master	24	Final deliverable of D1.1. With updated service descriptions and all sections complete

MPOWER Consortium

MPOWER (Contract No. 034707) is a *Specific Targeted Research or Innovation Project (STREP)* within the 6th Framework Programme, Priority 2.5.11 (eInclusion). The consortium members are:

<p>SINTEF ICT (Project Coordinator) NO-7465 Trondheim, Norway</p> <p>Contact person: Marius Mikalsen Email: marius.mikalsen@sintef.no</p>	<p>Ericsson Nikola Tesla d.d. Address</p> <p>Contact person: Ivan Benc Email: ivan.benc@ericsson.com</p>
<p>ARC Seiberdorf research GmbH Wien, Austria</p> <p>Contact person: Barbara Prazak Email: barbara.prazak@arcsmed.at</p>	<p>Psykiatrien i Vestfold HF Tønsberg, Norway</p> <p>Contact person: Torild Holthe Email: torhild.holthe@aldringoghelse.no</p>
<p>Uniwersytet Jagiellonski Collegium Medicum Krakow, Poland</p> <p>Contact person: Mariusz Duplaga Email: mmduplag@cyf-kr.edu.pl</p>	<p>TB-Solutions Advanced Technologies S.L. Zaragoza, Spain</p> <p>Contact person: Mayte Hurtado Email: hurtadom@tb-solution.com</p>
<p>University of Cyprus Nicosia, Cyprus</p> <p>Contact person: Eleni Themistokleous Email: eleni.themistokleous@cs.ucy.ac.cy</p>	<p>Dimension Informatica Valencia, Spain</p> <p>Contact person: Juan Jose Cubillos-Esteve Email: j.cubillos@dimension-informatica.es</p>

Table of Contents

Release History.....	3
MPOWER Consortium.....	4
Table of Contents	5
Table of Figures.....	8
List of Tables.....	9
1 Executive summary	10
2 Introduction	12
2.1 Role of this deliverable.....	12
2.2 Relationship to other MPOWER deliverables.....	12
2.3 Structure of this document.....	12
3 MPOWER Target System	14
3.1 Static Structure	14
3.2 System in Use	16
4 The MPOWER Framework.....	18
4.1 MDSD HealthCare Framework.....	18
4.2 MPOWER Architecture.....	19
4.3 MPOWER UML Extensions	19
4.4 MPOWER Middleware	19
4.5 MPOWER Applications	19
5 The MPOWER Development Domain: Actors and Assets	20
5.1 Development Actors (stakeholders)	20
5.2 Environment Actors (systems)	21
5.3 Domain Assets.....	22
5.3.1 Dictionaries.....	22
5.3.2 Standards	23
6 MPOWER Architecture and Platform.....	25
6.1 MPOWER Reference Architecture	25
6.1.1 Conceptual Service Model	25
6.1.2 Architectural styles and principles	26
6.2 MPOWER SOA Architecture.....	26
6.2.1 Conceptual Service Architecture	26
6.2.2 SOA Reference Architecture	27
6.3 MPOWER Information Models.....	29
6.3.1 Process of defining information models.....	30
7 MPOWER Middleware Services.....	32
7.1 Communication Services - (Logical diagram).....	32
7.2 Information Services - (Logical diagram)	33
7.3 Management Services - (Logical diagram)	34
7.4 Security Services - (Logical diagram)	34
7.5 Sensor Services - (Logical diagram)	35
8 MPOWER Methodology	37
8.1 Model-Driven Software Development	37
8.1.1 OMG's Model Driven Architecture (MDA)	38
8.1.2 Model Transformation and Code Generation	38
8.1.3 Meta-models, UML Profiles and UML Patterns	39
8.2 SOA, MDSD and HealthCare - a way to improve the systems' compliance to standards.....	40

8.2.1	Conceptual Model	40
8.2.2	Create UML Profile and Model Transformation from Healthcare Standard.....	40
8.2.3	Create Healthcare Middleware Service using UML Healthcare Profile.....	41
8.3	MDA Tool Support for Specifying Services in MPOWER.....	43
8.3.1	Information Modelling – HL7 standard messages.....	44
8.3.2	Modelling: Computation Independent Models - User needs.....	44
8.3.3	Modelling: Platform Independent Models - Service Specification	46
8.3.4	Modelling: Platform Specific Models - WSDL Models and Code.....	46
8.3.5	The MPOWER Tool Chain	47
8.3.6	Tool Chain Example.....	48
8.4	Where MDA should be used in MPOWER.....	49
9	MPOWER Application Platform.....	50
9.1	MPOWER Recommended Deployment Platform.....	51
9.1.1	Application Server.....	51
9.1.2	Business Process Execution	52
9.1.3	Databases and Data Access	53
9.1.4	Messaging.....	53
9.1.5	Communication networks.....	53
9.1.6	Firewall Issues	53
10	Related work.....	54
10.1	Healthcare Service Specification Project (HSSP)	54
10.2	Open Healthcare Framework.....	54
	References	55
Appendix A	IBM Reference Architecture SOA	56
A.1	An architectural template for SOA.....	56
A.2	IBM SOA Architectural template and MPOWER Platform.....	57
A.3	Modelling styles using this reference architecture	57
A.3.1	Component (or Service component or Enterprise component)	58
A.3.2	Composite (or Composite services).....	59
Appendix B	UML Profiles for Homecare.....	60
B.1	Introduction	60
B.2	Background and Related work	61
B.3	Methods	61
B.3.1	Activity 1: Capture Domain Knowledge.....	62
B.3.2	Activity 2: Designing a Toolchain for MDD in Homecare	62
B.3.3	Activity 3: Refine the MPOWER toolchain and develop a DSML.....	63
B.4	Results	63
B.4.1	Activity 1: Conceptual Domain models	63
B.4.2	Activity 2: The MPOWER Toolchain	64
B.4.3	Activity 3: Refined Toolchain - Mapping of Domain Concepts to DSML - UML Profile	65
B.4.4	Homecare UML profile	65
B.4.5	SOA Homecare UML Profile	66
B.5	Discussion	68
B.6	Concluding remarks.....	69
Appendix C	MDSD and Interoperability	72
C.1	Abstract:	72
C.2	Introduction	72
C.3	Immature MDSD tools and need for evaluations	73
C.4	Many Systems, Many Standards	74
C.5	A MDSD Framework for HealthCare	74

C.6	Results	74
C.6.1	Example of MDSD Healthcare Framework in Homecare	74
C.6.2	A Simple UML Profile for HomeCare	75
C.6.3	The Healthcare Information Systems	75
C.7	Discussion	77
C.8	Future work	78
C.9	Conclusion.....	78
C.10	References	78
Appendix D	Reusable Actors and Services.....	80
D.1	Introduction	80
D.2	Methods and Materials	81
D.3	Results	81
D.4	Discussion	83
D.5	References	83

Table of Figures

Figure 1: Instance of a MPOWER Target System	15
Figure 2: Example of a Typical MPOWER-based Smarthome - Call Center system	16
Figure 3: The MPOWER Technological Overview	18
Figure 4 Overview of development actors	20
Figure 5 SOA Conceptual Model	25
Figure 6: Dynamic SOA as described in SOA4HL7 Architecture document [7].....	27
Figure 7: The MPOWER SOA Reference Architecture.....	29
Figure 8 Process of defining information models	30
Figure 9: MPOWER Service Categories	32
Figure 10: Detailed view of the communication services	33
Figure 11 Detailed view of the information services	33
Figure 12 Detailed view of the management services	34
Figure 13 Detailed view of the security services.....	35
Figure 14 Detailed view of the sensor services	36
Figure 15 : The MDA models. Figure based on [22]	39
Figure 16: High-level conceptual process model	40
Figure 17: Creating a UML Healthcare profile and Model transformation to support a Healthcare Information Standard	41
Figure 18: Creating reusable Healthcare Middleware Services using Healthcare UML Profile and Model Transformations	42
Figure 19: Creating a Healthcare Application using Healthcare UML Profile, and Model Transformation and reusing Middleware Service(s)	43
Figure 20: Use Cases for Management scenarios.....	45
Figure 21: Use Case "Stakeholder management" and the related scenarios.....	45
Figure 22: Features derived from Stakeholder management and Add plan element use cases	46
Figure 23: Service rationale. The MedicationManagment service implements five features .	46
Figure 24: The Service Model with UML Stereotypes	46
Figure 25: The MPOWER Tool Chain and Artefacts	48
Figure 26: WSDL model for ActorManagement Service.....	49
Figure 27: Example of MPOWER system deployment.....	50
Figure 28 IBM SOA Reference Architecture	56
Figure 29 MPOWER Reference Architecture	57
Figure 30 Cooperation between services and enterprise components	58
Figure 31 Explanation of component in SOA story	58
Figure 32 Way of building composite services	59
Figure 33 The three main project activities and their work products	62
Figure 34 Diagram showing the main concepts in a smart homecare domain	64
Figure 35 The Service-oriented view on a typical homecare environment	64
Figure 36 First version of Homecare UML Profile	66
Figure 37 The Homecare SOA UML Profile diagram	68
Figure 38: A subset of the CONTSYS CarePlan concept	75
Figure 39: The Care Center System PIM	75
Figure 40: The GP EHR Homecare Extension PIM.....	76
Figure 41: Java PSM for the Care Center System	76
Figure 42: Java PSM for the GP EHR HomeCare Extension.....	77
Figure 43: The iterative model-driven development process used to identify actors and services. Artefacts are shown as rectangles whereas the activities are denoted as rectangles with rounded corners.	81

Figure 44: The ActorModel showing the elements of the system, stakeholder and role parts. 82
Figure 45: The relationships between healthcare professional actors and roles as defined in the ActorModel 82
Figure 46: Actors, Roles and Use cases in the usecase diagram for Calendar Management services 82
Figure 47: The service categories in the ServiceModel. The elements of each category are reusable services 83

List of Tables

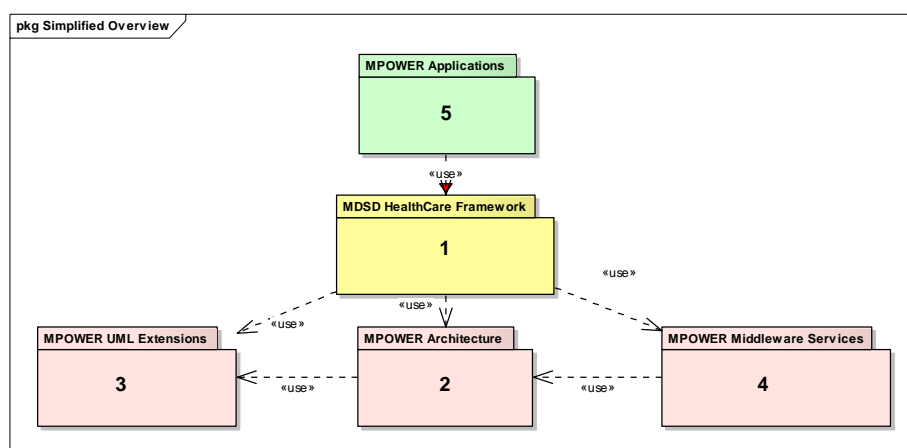
Table 1: Different types of messages exchanged in healthcare 24
Table 2: Platform artefacts and implementation technologies 51
Table 3. Table describing the proposed stereotypes in and tagged values in the Homecare UML profile 65
Table 4. List of stereotypes and tagged values in the SOA Homecare UML Profile 67

1 Executive summary

A core deliverable from the MPOWER Project is the architectural specifications of the services and components that will enable developers to rapidly create new and interoperable applications to support elderly and cognitive disabled in a smart-house environment.

The technological components of the MPOWER solution are comprised of four core parts: 1) MDSD (Model-Driven Software Development) Healthcare Framework, 2) MPOWER Architecture, 3) MPOWER UML Extensions, and 4) MPOWER Middleware. In addition, MPOWER Applications (5) will be developed using the MDSD Healthcare Framework and the MPOWER Middleware.

- The MDSD Healthcare Framework (1) has guidelines for the middleware and application development process, tool selection and configuration. Focus is on Model-Driven Software development tools and techniques. The MDSD approach is described in a separate chapter.
- The MPOWER Architecture (2) builds on the concept of Service-Oriented Architectures (SOA). More specifically, MPOWER uses the reference architecture for SOA specified by IBM to structure the services and components that are a part of the MPOWER Middleware (4).
- The MPOWER UML Extensions part (3) provides a formal MPOWER domain specific language to (1) and (2) to represent the concepts and services in the form of two UML Profiles.
- The MPOWER Middleware is a collection of reusable services and components, specified in UML with extensions (3) following the structure given in (2) and the guidelines in (1).



Based on user scenarios and needs, five categories of services are included in the MPOWER Middleware (4). The categories are:

- **Communication Services:** services for sending messages and notifications to users and systems.
- **Information Services:** services setting and getting information for individual plan, calendar, medication list, and knowledge sources.
- **Management Services:** services for managing services, users, access rights and system contexts

- **Security Services:** services for authentication and authorization of users and system components.
- **Sensor Services:** services for configuring (add, remove, adjust) devices and retrieving sensor information.

Four appendices are included in the deliverable: a description of the IBM SOA Reference Architecture, a scientific paper from MOTHIS 2008, presenting the UML Profiles developed to extend the UML core with homecare concepts, a scientific paper from MEDINFO 2007 presenting the MPOWER MDSD and interoperability concepts, and a scientific paper from MIE 2008 presenting the reusable Actors and Service defined in MPOWER.

2 Introduction

2.1 Role of this deliverable

The purpose of this document is to specify the MPOWER Overall Architecture. This architecture describes all architectural aspects of an MPOWER-based information system, including guidelines for applying the architecture.

The document should enable architects and developers to

- Understand which type of systems that can be built using the MPOWER middleware
- Define which actors that are involved in the development process
- Document the rationale for the middleware building blocks based on the user scenarios, usecases and features developed in WP7
- Understand and use the MPOWER Reference Architecture
- Understand the SOA concepts and usage of MPOWER middleware building blocks
- Understand how to implement and add building blocks to the MPOWER middleware
- Understand how to deploy and run MPOWER services

The document is *not*:

- A documentation of a specific MPOWER-based information system

To accomplish the defined purposes, the document describes not only the proposed Reference 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

- Will apply the MPOWER middleware in the design and / or development of a software system
- Will specify or update the MPOWER middleware
- Need to understand the architecture of a MPOWER based software system

2.2 Relationship to other MPOWER deliverables

- D.1.2 Developers Handbook. D1.2 provides details about the development process
- D.1.3 Service LifeCycle model: D1.3 provides details about the use and maintenance of the services in MPOWER
- D7.1 User scenarios and needs: The basis for the service identification
- Service design documents (D2.2, D3.1, D3.2, D4.2 and D5.2) are all based on the MPOWER architecture.

2.3 Structure of this document

This chapter has given a brief overview of the role of this document, and its relation to ther deliverables in MPOWER.

In Chapter 3, a technical overview is provided, giving an overview of the main parts of MPOWER and how these are related. It includes a presentation of the architecture and

middleware which is the foundation for the MPOWER applications, but also the MPOWER UML Extensions and MDSD framework which will guide and simplify the development of healthcare applications.

Chapter 5 the actors and artefacts involved in development of MPOWER middleware, services, and applications are presented, including development actors and artefacts as well as environment systems and artefacts such as dictionaries, standards and patterns.

Chapter 6 presents the architecture of MPOWER in more detail. The reference architecture as well as MPOWER's SOA architecture is described.

In Chapter 7 a further description of the MPOWER middleware is presented, including the main categories of services defined: communication, information, management, security, and sensor services.

Chapter 8 gives an introduction to how model driven architecture will be applied in MPOWER. It includes an introduction to MDA, motivation for why to MDA and how it can improve the development of healthcare applications and more details on how an MDA tool-chain can be established for MPOWER.

Chapter 9 presents the MPOWER application platform, focusing on concrete technologies and implementations.

Chapter 10 describes two related initiatives, namely the HSSP project and the Open Healthcare Framework initiative.

The appendixes present further reference and background material. In Appendix A the IBM SOA reference architecture, on which the MPOWER reference architecture is built, is presented in more detail. Appendix B presents the UML profiles that extend the core UML specification with smart homecare information. Appendix C describes the overall concept of using MDSD to improve interoperability of healthcare information systems. Finally, Appendix D describes how central modelling constructs such as actor models and service models can be reused across project, organizations and national borders to further improve adherence to standards and thus interoperability.

3 MPOWER Target System

This document describes the overall architecture of a distributed healthcare information system based on the MPOWER development principles and running on a MPOWER defined platform. In other words, the *target system*, as defined in IEEE 1471:2000, includes services and components that are specified and reused from the MPOWER project.

3.1 Static Structure

This section will define the target system in terms of MPOWER Services and the MPOWER Service domain; smart homes that support elderly and cognitive disabled.

Four concepts are relevant:

- Smart Home: the residence of a user that needs assistance from smart sensors and a supporting organization.
- Care Center: A supporting organization responsible for managing and providing care and support to a *Subject of Care* – the person receiving care and assistance.
- MPOWER Service Platform: a set of software services and components offering secure interfaces to access local and central communication and information services. The interfaces use the HL7 messaging standard as a syntactical and semantical platform to the other instances of the MPOWER Service Platform and MPOWER Common Services.
- MPOWER Common Services: a set of shared and reusable information and communication services enabling the development and deployment of smart home care solutions.

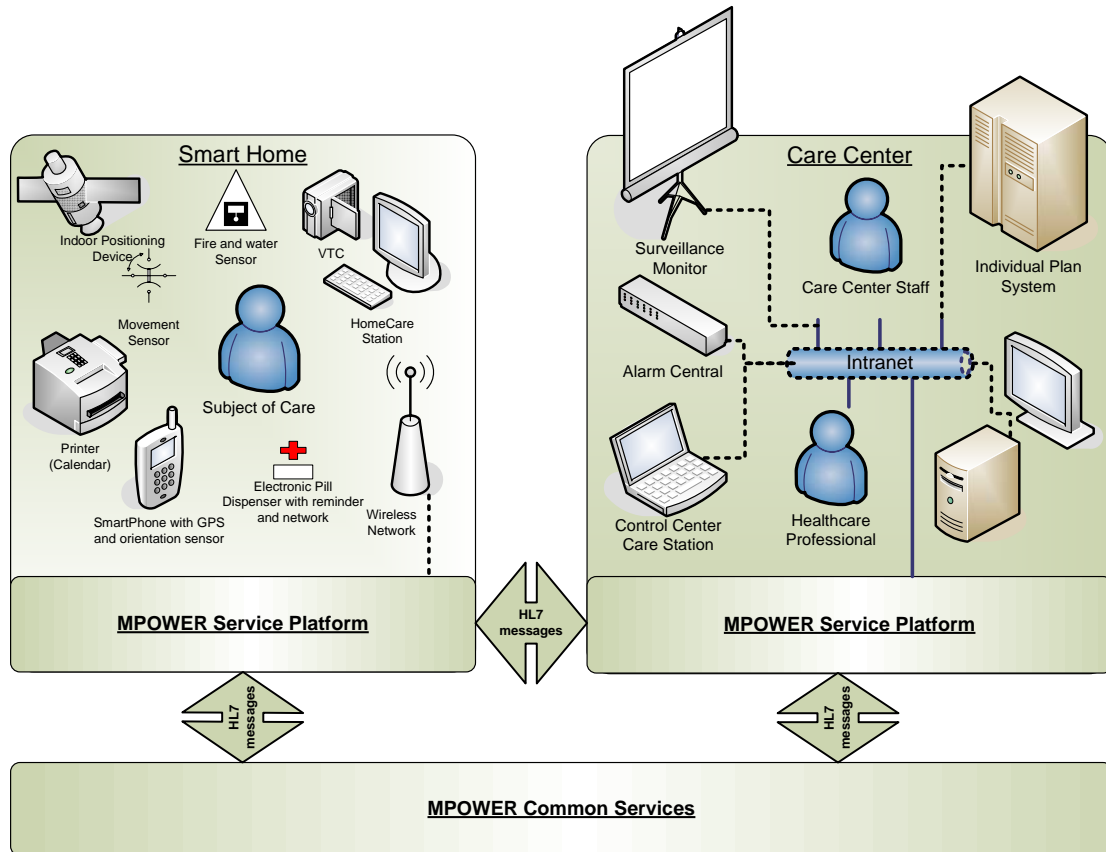


Figure 1: Instance of a MPOWER Target System

Figure 1 shows an instance of the MPOWER Target System where a Smart Home is connected to a Care Center through the MPOWER Service Platform and the MPOWER Common Services. From the Care Center, the staff and care personnel can monitor and respond to sensor alarms from the Smart Home. Services like video teleconferencing (VTC) and positioning is supported by the MPOWER Service Platform. Access control, messaging and information sharing are typical services to be provided by the MPOWER Common Services.

Figure 2 shows a simplified software service structure of the system shown in Figure 1. In accordance with the Service Oriented Architecture concept[1, 2], the MPOWER services are specified as **self-contained** services that are available across the enterprise that can be evoked through standard protocols both internally and externally. In MPOWER, the standard message format used for service interfaces is HL7 v3 messages. Figure 2 shows that the MPOWER Service Platform at the Smart Home consists of one service called Sensor Service. The HomeCare Application uses this service and external services located at the Healthcare Network Services and Care Center nodes. The information is transferred using HL7 messages.

The External Notification Service is deployed as a MPOWER Common Service and shared by the Healthcare Network Services organization. CalendarManagement and CalendarSynchronizer services are deployed to a server at the Care Center and exposed externally through the CalendarService interface.

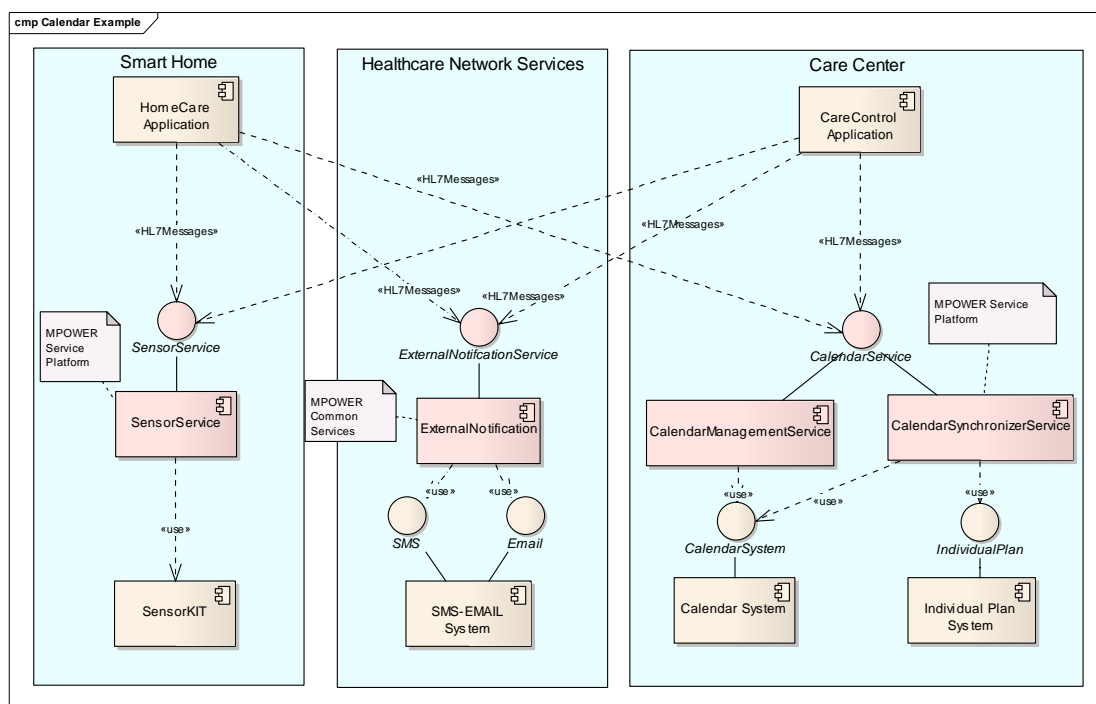


Figure 2: Example of a Typical MPOWER-based Smarthome - Call Center system

3.2 System in Use

Figure 1 and Figure 2 show examples of a MPOWER-based system in a distributed environment where the subject of care (also known as the patient) and the caregivers are separated geographically. This is opposed to a nursing home or hospital solution where both are in the same building (-complex) and traditional information systems concepts are applicable.

The main functions offered by a MPOWER-based system are:

- Information sharing between Care Center and Smart Home:
 - Medication and Calendar information: patients, caregivers and family members have access to updated medication and calendar information. The level of access can be configured for each group. Reminder functionality is a key feature

- Messaging: the ability to securely send messages between patients, caregivers, family members and other registered stakeholders in the system
- Education material and Q&A: people suffering from dementia can get assistance from information systems that provide education material and guides for a variety of everyday problems and activities.
- Monitoring of Smart Home from Care Center
 - Safety sensors with alarming: domotic sensor networks such as water flow monitors, motion detection and temperature sensors is installed in the patient's home to improve the safety and security for the patients and their family.
 - Physiological monitoring of patient: patients requiring medical surveillance can connect physiological sensors such as pulseoximetry measurement devices to the smart home system. The measurements will be captured, stored and transferred to the care center.
 - Location and position sensors: in some cases, outdoor and indoor location sensor systems can be installed to monitor the movements of a patient. This is especially relevant for people with dementia.
- Flexible configuration of system characteristics
 - Easy configuration of new sensors and information systems: for patients with dementia, the required level of ICT support will change over time. Therefore the system provides flexible configuration mechanisms, allowing new services to be added easily, both before and after first installation in the patient's home.
 - Context aware monitoring: to tailor the "smartness" of the smart home care system, the system provides a configurable context manager service that specifies the system behaviour based on the status of context variables such as sensors, time and personal requirements.

4 The MPOWER Framework

The technological components of the MPOWER solution are comprised of four core parts:

- 1) MDSD (Model-Driven Software Development) Healthcare Framework
- 2) MPOWER Architecture
- 3) MPOWER UML Extensions
- 4) MPOWER Middleware

In addition, MPOWER Applications will be developed using the MDSD Healthcare Framework and the MPOWER Middleware (e.g. the proof of concept applications developed in MPOWER work package 6).

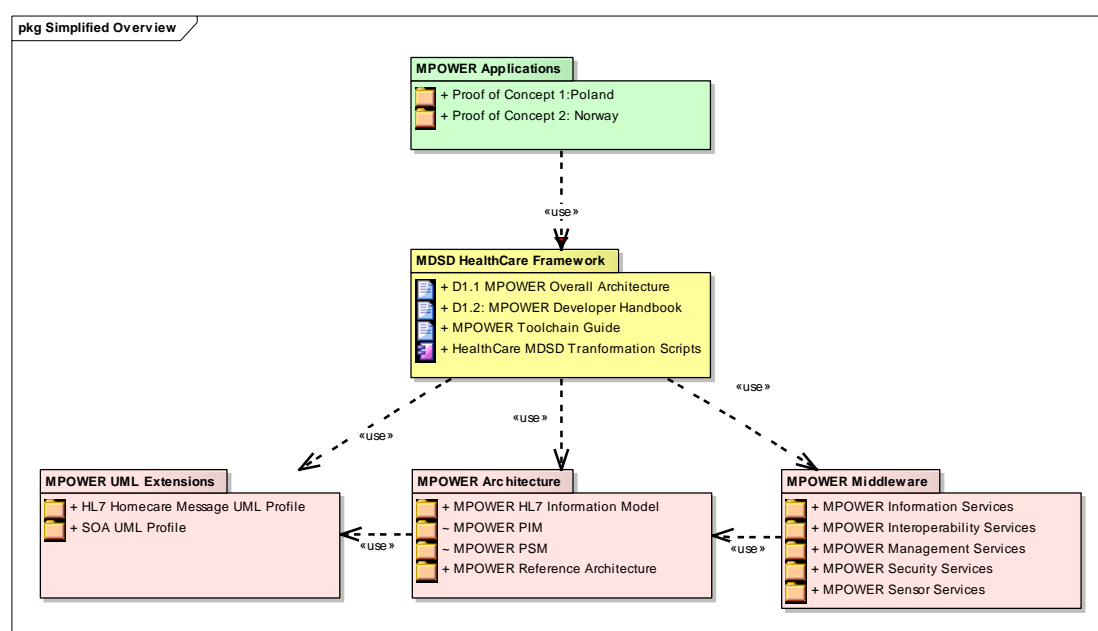


Figure 3: The MPOWER Technological Overview

The four core parts of MPOWER and the applications are shown as packages illustrated in Figure 3, and are all related through use and dependency relationships. The packages are described further in the following subsections.

4.1 MDSD HealthCare Framework

The MDSD HealthCare Framework provides guidelines for the middleware and application development process, tool selection and configuration.

- Using the concepts from MPOWER UML Profiles, model transformation scripts will create Platform Specific Models (PSMs) and code that have attributes, operations and optionally some supporting classes, required for interoperable information exchange between systems based on the international standards supported by MPOWER
- Using the Reference Architecture from the MPOWER Architecture, it provides guidance to designing MPOWER-based applications.
- Using the MPOWER Middleware, it provides structure and pattern templates for designing MPOWER-based applications

4.2 MPOWER Architecture

The MPOWER Architecture package consists of the Reference Architecture, MPOWER HL7 Information models, and UML models that specify reusable services and components. Using UML Patterns defined in the MDSH Healthcare Framework, and Profiles defined in the MPOWER UML Extensions, domain-specific and technology independent UML models are described as Platform Independent Models (PIMs). The PIMs can be transformed into PSMs adding platform specific mappings using the transformation scripts described in the framework.

4.3 MPOWER UML Extensions

The MPOWER UML Extensions provide standards-based UML Profiles (stereotypes, tagged values and constraints) and model transformation scripts. The UML Profiles define concepts based on international standards such as HL7 and CEN TC251 EN12967/13606 incorporating the standards into the design of the software system. The MPOWER project focuses primarily on HL7 messaging, but the concept of using UML extensions can be reused for other standards as long as they are specified at a sufficient degree of formality. Two UML Profiles have been specified in the project, namely the Homecare UML Profile and the SOA Homecare UML Profile as presented in Appendix B.

4.4 MPOWER Middleware

The MPOWER Middleware holds reusable and compiled (runnable) services and components that can be easily utilized by application developers. The MPOWER services and components are organized into five functional categories; information services, interoperability services, management services, security services and sensor services. They are structured according to the packages and layers defined in the MPOWER Reference Architecture described in the MPOWER Architecture Package. The reusable services and components are implementations of the Platform Independent Models (PIM) and Platform Specific Models (PSM) in the MPOWER Architecture. Using the provided model transformation and the guidelines in the Developer Handbook, new platform specific services and components can be added to the MPOWER Middleware as new technological platforms are being introduced.

4.5 MPOWER Applications

Two Proof-of-Concept applications have been developed using the MPOWER Middleware services as the core artefacts. To guide the developer, the MDSH Framework (the developer handbook) provides a description of which tools that should be used and how to configure them. It also provides a guideline for how to develop new generic and reusable middleware services and components to be included in the MPOWER Middleware.

5 The MPOWER Development Domain: Actors and Assets

5.1 Development Actors (stakeholders)

The following is a brief overview of the different development actors involved in the development, installation and management of the MPOWER platform and application. The different actors are described further in the D1.3 Service Lifecycle Model document.

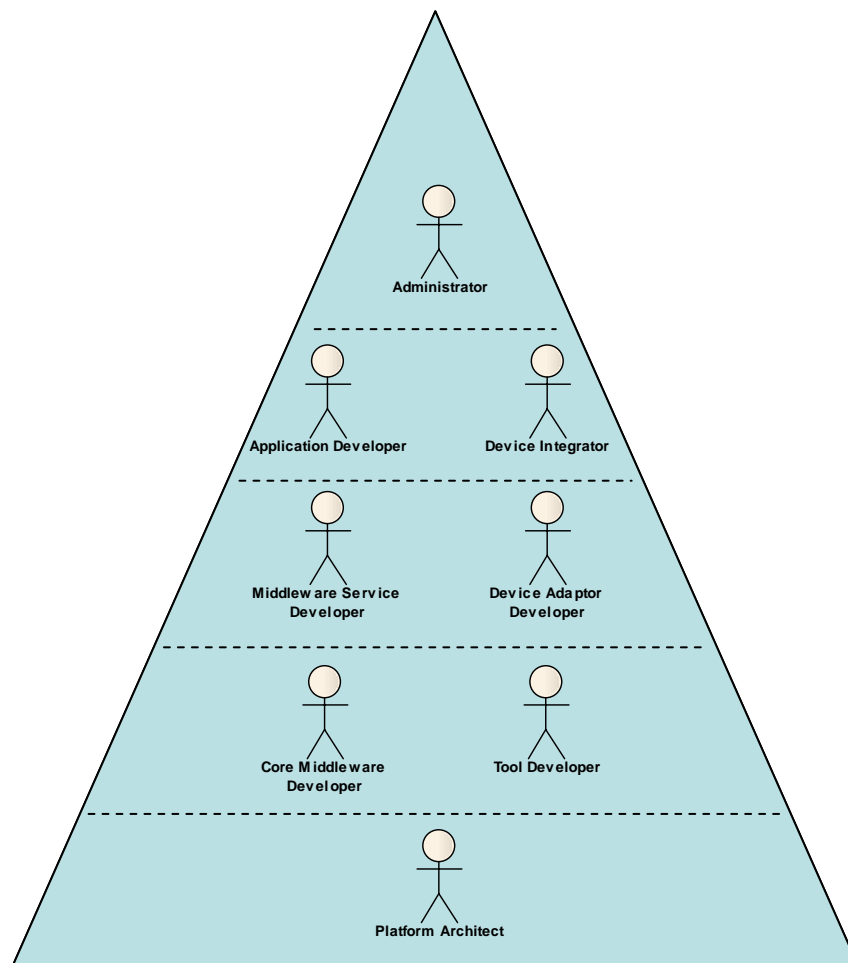


Figure 4 Overview of development actors

- The Platform Architect develops the service architecture, identifies services and generates service specifications for the middleware. Together with the system owner and national standardization bodies (not included in this hierarchy), the architect will choose architectural styles and standards to conform to. A popular style that is the core of MPOWER, is the Service Oriented Architecture.
- The Core Middleware Developer develops the generic core of the middleware. The core provides the minimum set of services and components required to implement functional MPOWER Services.

- The Tool Developer develops the MSD toolchain which will enable the other developer roles to more efficiently develop services and applications based on the MPOWER middleware.
- The Middleware Service Developer will use the core middleware services to implement MPOWER services. MPOWER services are specific for the smarthome and homecare domains and provide valuable functionality to application developers.
- Application Developers will use the MPOWER services to develop end user applications for the smarthome and/or homecare domain.
- The Device Adaptor Developer creates adaptors which enable a device / set of devices to work with the middleware.
- The Device Integrator will install and set up devices for integration with the middleware services.
- The Administrator will install the platform on servers and user site. The administrator will also manage the applications, user rights and handles system exceptions.

5.2 Environment Actors (systems)

When designing, developing, deploying or maintaining systems for homecare it is of utmost importance to identify and understand the homecare system's relation to *environment actors*. Typical *environment actors* include information systems, sensor systems and communication systems. The homecare system will interface these systems and it is important to document the details about why and how.

Environment actor categories:

- Information Systems: Systems that provide information and / or information services through a defined interface. Typical information systems are patient administrative systems, calendar systems, medication systems or systems that provides services for verifying the structure and / or validity of an information element.
- Sensor Systems: Systems that provides measured information through a defined interface. A sensor can be both physiological and non-physiological. Automation services rely on sensor information from e.g. door sensors, water-temperature sensors, light sensors and movement sensors. A sensor system can be composed of several sensor-s/systems).
- Communication Systems: To communicate information from the user's home, a communication system must be used. Typical networks include fixed networks (xDSL), LAN, WLAN, GSM/GPRS, PSTN. The communication system may require special protocols, formats and security mechanism.
- Online information resources: This includes sources of information that are available on the Internet for free or through a subscription. These information sources are different from the "Information Systems" category on several points including: reliability, ownership, accessibility and access mechanisms (normally an Internet Browser such as Opera, Firefox, Safari or Internet Explorer).

For each *environment actor* the following characteristics should be described:

- Type of actor: according to the categories defined above

- **Owner:** The owner of the *environment actor*. Typically this means the person or organization that must be contacted in order to access the services or information that the *actor* provides.
- **Use pattern:** The situations in which the *environment actor* can / should be used. Can be described using pre-condition, post-condition and invariants. A UML Sequence diagram or Activity diagram can be clarifying.
- **Interface details:** The protocols required to access the *Environment Actor* such as network protocols, messaging protocols and possibly security mechanisms. If relevant, operations on the interfaces (e.g. Web Service WSDL) should be described or referenced.
- **Where to find more information:** The main location for acquiring more information. Preferably an organization or person. A website reference is insufficient since it may be changed on a long term.

5.3 Domain Assets

In section 5.2 the categories of *environment actors* were described. These actors use *artefacts* such as dictionaries, standards, patterns, components or documents. In this section a list of typical artefacts relevant for the development of a architecture is provided.

5.3.1 Dictionaries

The healthcare domain involves a plethora of professions and cultures, each having their own way of expressing different phenomena, situations or concepts. This often leads to misunderstanding and misinterpretations of communication across professions, departments and cultures.

There are many ways to overcome lack of a common terminology and understanding of concepts.

- **Thesauri:** A *thesaurus* is a list of terms used for a certain application or domain. A thesaurus is intended to be complete for its domain and can also contain a list of synonyms for each preferred term. A thesaurus is also called a “controlled vocabulary”
- **Nomenclatures** A *nomenclature* assigns codes to medical concepts, and medical concepts can be combined according to specific rules to form more complex concepts. This leads to a large number of possible code combinations.
- **Ontologies** A common way to express a dictionary in a domain is to use an *ontology*. Gruber¹ has defined ontology as: “An *ontology* is a formal explicit specification of a shared conceptualisation”. A conceptualisation, in this context, refers to an abstract model of how people think about things within the healthcare domain. Since the healthcare domain is so large, there are many ontologies available, each covering one or more parts of the domain.
- **Coding:** In healthcare, coding of e.g. medical diagnosis, disease, and medication is a way to create a classification. There are a large number of classification systems, some overlapping. Thus, an architect working in the healthcare domain must chose between classifications according to context and business.

¹ Gruber, T. “A translation approach to portable ontologies”. *Knowledge 7*.

5.3.1.1 How to use dictionaries in MPOWER

It is necessary to use a dictionary when integrating information from semantically heterogeneous sources. The original information sources must be related to the dictionary according to certain rules or translations in order to provide a common information model for the integrated system. The mapping or translation can be done using standard patterns for information integration.

5.3.2 Standards

A standard is a formalised model or example developed by a standardisation organisation or established by general consent. Using standards help achieving interoperability with other systems following the same standard, thus simplifying a future integration of information from these systems.

There are many types of standards to be followed when developing distributed information systems. Some standards are international while others are only valid on a national level, e.g. legislations regarding documentation of work done by healthcare providers. For this reason, this document will point to international, European and Norwegian standards and standardising institutions.

Standards relevant for the MPOWER Platform are identified and described in [3-5]

5.3.2.1 Legislations and regulations

These are standards that must be followed – either by implementing the system according to the standards, or by avoiding it by redefining the system's capabilities or extent. The national legislative assembly gives legislations and regulations. Often, a dedicated institution is empowered to make sure that these are followed.

5.3.2.2 Information representation standards and models

There is a large work going on trying to standardise the information models within the healthcare domain. The work is complex and coordination across country and culture borders is difficult. The MPOWER system solution seeks to simplify the selection and use of these standards by providing a guidelines and tools that incorporates concepts from the standards. The architect and developers will only need to consider which standard to use in the system, and select the appropriate tooling support from the MPOWER system solution. Below is a list of international standards that are relevant for MPOWER architectures.

International information standards:

- **CEN TC251.** European Standardization of Health Informatics - Standardization in the field of Health Information and Communications Technology (ICT) to achieve compatibility and interoperability between independent systems and to enable modularity. This includes requirements on health information structure to support clinical and administrative procedures, technical methods to support interoperable systems as well as requirements regarding safety, security and quality: <http://www.centc251.org>
- **Health Level 7 (HL7).** Health Level Seven is one of several ANSI-accredited Standards Developing Organizations (SDOs) operating in the healthcare arena. Most SDOs produce standards (sometimes called specifications or protocols) for a particular healthcare domain such as pharmacy, medical devices, imaging or insurance (claims processing) transactions. Health Level Seven's domain is clinical and administrative data: <http://www.hl7.org>

- **OpenEHR.** The openEHR Foundation is a non-profit organisation bringing together an international community of people working towards the realisation of clinically comprehensive, ethico-legally sound and interoperable electronic health records to support seamless and high quality patient care: <http://www.openehr.org>

5.3.2.3 Messages exchange standard

The environment of medical electronic data interchange (EDI) is extremely heterogeneous and complex, and it is changing continuously. Bommel and Musen have created a list of different types of messages exchanged in healthcare:

<p>Clinical messages</p> <ul style="list-style-type: none"> - Exchange of service requests to and reports from laboratories, radiology departments, and ancillary services - Prescriptions from physicians to pharmacies - Hospital admission data and discharge summaries - Multimedia patient-centred electronic health care records - Transplantation data, such as registrations, waiting lists, and organ matching - Data from pharmaceutical industry, e.g., information on drugs, drug surveillance, and pharmaceutical trials - Interpersonal mail between practitioners, e.g., between general practitioners and specialists - Information retrieval from external literature and knowledge bases - Communication with public authorities in connection with epidemiology, quality assessment schemes, or utilization review <p>Logistics and financial messages</p> <ul style="list-style-type: none"> - Communication between hospitals and suppliers; purchasing, invoicing, and logistics - Exchange with insurance agencies and third-party payers; billing and reimbursement <p>Medical images, biosignals, and multimedia data</p> <ul style="list-style-type: none"> - Multimedia patient record - Conventional X-ray images from radiology departments - Digital images from computed tomography scanners, magnetic resonance imagers, and ultrasound equipment - Images processed for radiotherapy and neurosurgery - Scanned documents (e.g., for the electronic multimedia patient record) - Digital voice reports - Biosignals (electrocardiograms, electromyograms, electroencephalograms, etc.)
--

Table 1: Different types of messages exchanged in healthcare

6 MPOWER Architecture and Platform

This chapter presents the reference architecture user for MPOWER-based systems. A reference architecture is defined as:

“A high-level, generic architecture which is used as the basis for development of concrete system architectures, and to compare architectures of existing systems to each other.” [MPOWER Glossary]

The main purpose of using a common reference architecture in MPOWER is to aid the developers (application and middleware) in the definition of services; use the same abstraction level and concepts for interaction.

6.1 MPOWER Reference Architecture

The MPOWER project uses IBM Service Oriented Architecture (SOA) as reference architecture [6]. The next chapters briefly describe the main element of IBM SOA reference architecture and the relationships between them.

It starts with a conceptual model that describes the main interaction between three SOA primary parties and then continues with description of IBM architectural styles and principals.

6.1.1 Conceptual Service Model

The main parties that are involved in service-oriented architecture are (Figure 5):

- **Service Provider:** provides the description and implementation of a service
- **Service Consumer:** can either use the uniform resource identifier (URI) for the service description directly or can find the service description in a service registry and bind and invoke the service
- **Service Broker:** provides and maintains the service registry. The next diagram shows the main interaction between the above mentioned parties

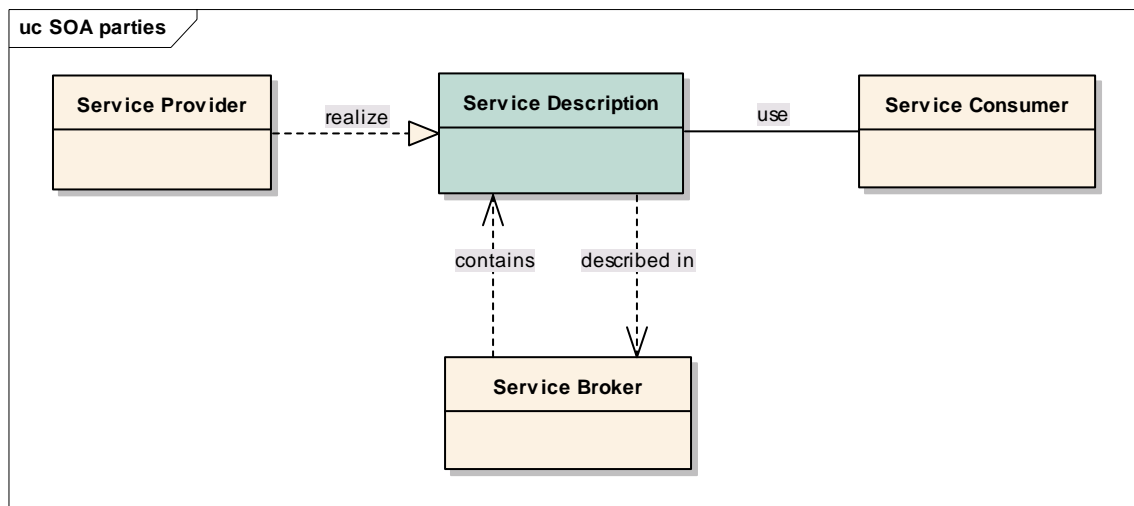


Figure 5 SOA Conceptual Model

6.1.2 Architectural styles and principles

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 know as components (some of them are available “off the shelf” and some are build 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.” [6]

6.2 MPOWER SOA Architecture

6.2.1 Conceptual Service Architecture

The SOA4HL7 architecture document specifies three different SOA topologies: minimum SOA, Mediated SOA and Dynamic SOA [7]. The conceptual SOA topology to be used for MPOWER based systems is the Dynamic SOA as defined in the SOA4HL7 document, which involves dynamic use of a number of supporting platform services/components. These services, such as service management, policy management and orchestration manager comes as an integral part the chosen application server. The recommended servers for use with MPOWER are described in section 9.1.

Figure 6 describes the conceptual service architecture as described in the SOA4HL7 architecture document. With reference to Figure 6, two main groups of components are described: the core components and a set of supporting components. The core components are mandatory, whereas the supporting components are optional. However, many of the supporting components should be provided to have a complete SOA system. The mandatory components are: *Services Registry, Service Consumer, Service Provider, Adapters and Routing and Transformation Intermediary*. Details about the optional components can be found in [7] and [1].

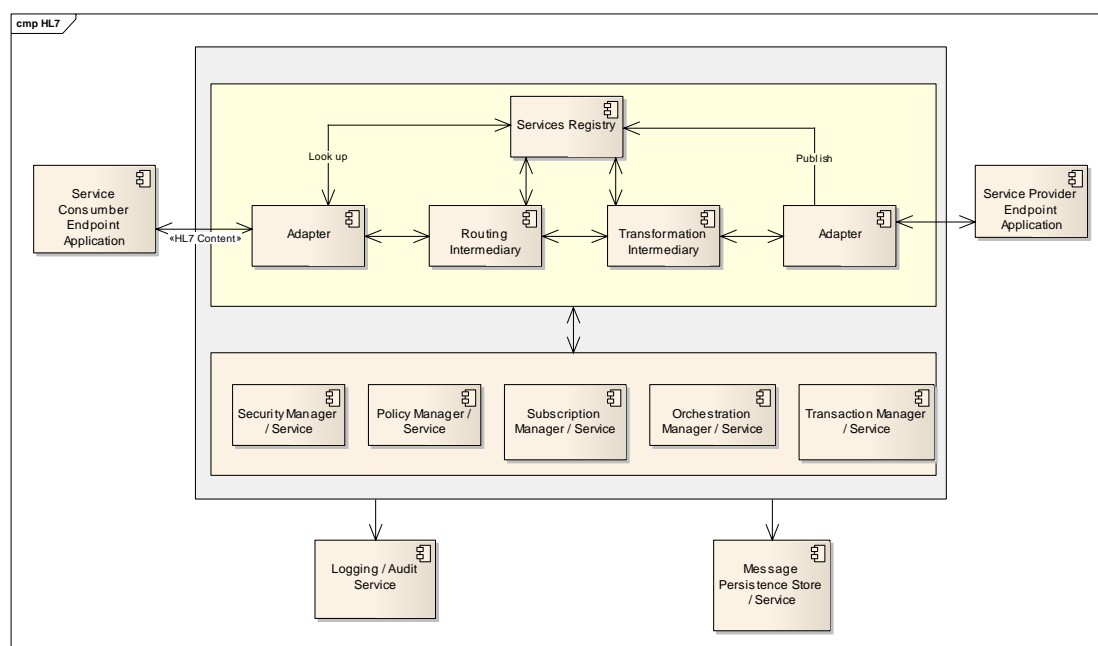


Figure 6: Dynamic SOA as described in SOA4HL7 Architecture document [7]

6.2.2 SOA Reference Architecture

The MPOWER SOA reference architecture is based on IBM's reference architecture for Service-Oriented Architecture[6]. 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 layers are grouped into three 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 Proof-of-Concept applications being developed in MPOWER), for the domain (smart house and homecare is the target domain for MPOWER services), and underlying systems (national health networks, vendor-specific databases and sensor systems). The groups and layers are described in the following.

6.2.2.1 Application Specific Components

Application specific components are considered to be designed for a specific type of use in an identified organization. Reusability across organizations/applications is not the primary concern when designing these components.

The **Application Layer** is usually out of scope for discussions on SOA, but it is useful to include it in the reference architecture to demonstrate which layers that will be included in a complete SOA system. The application layer need to provide a user interface that uses the underlying (business) services. It is important to note that SOA decouples the user interface from the components. The application layer will normally include components that are specific for the system (solution) being developed.

The **Business Process Layer** includes compositions and choreographies of services exposed by the Service Layer. The 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, and are to a certain degree application specific even though standardization on business processes is under development in many countries and medical specialities. An example of a homecare business process is management of a shared calendar where calendar, user (patient and caregiver) information, and medical plans are accessed through a set of services and service components.

6.2.2.2 Domain Specific Components

The domain specific components are specific for the operational domain, e.g., smarthouse solution and homecare systems. The services are designed according to best practice SOA principles with focus on reusability and loose coupling.

The **Services layer** includes 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 (business) service. The underlying service components provide service realization using the functionality provided by their interfaces. The interfaces get exported out as service descriptions in this layer, where they are exposed for use.

6.2.2.3 System Specific Components

System specific components are independent of both application and domain and can be reused by many applications in different domains. However, they are strongly coupled with the underlying system, being a software component or a hardware interface. Typical examples are sensor interfaces and database connections. Communication devices are relevant in distributed systems.

The **Service components** expose the functionality of the components and services in the resource layer. The Service components provide a high-level access to their information and control functions. A typical service component in MPOWER is a smarthouse-sensor driver that encapsulates and implements the sensor communication logic for the higher-layer services.

The **Resource Layer** consists of existing custom built applications, such as databases storing e.g., patient-administrative information, medication information, management information such as calendar events. Other relevant resources in MPOWER are information from (smart) sensors such as physiological monitoring devices, temperature sensors, burglar alarms and water-flow systems.

In Figure 7, the MPOWER reference architecture is shown in a UML Package diagram. Two categories of applications have been defined in the application layer: Homecare and Smarthouse. The architecture is not restricted to these types of applications, but the service-categories specified in the Service Layer are targeted towards these types (see chapter 7. for more details on service categories)

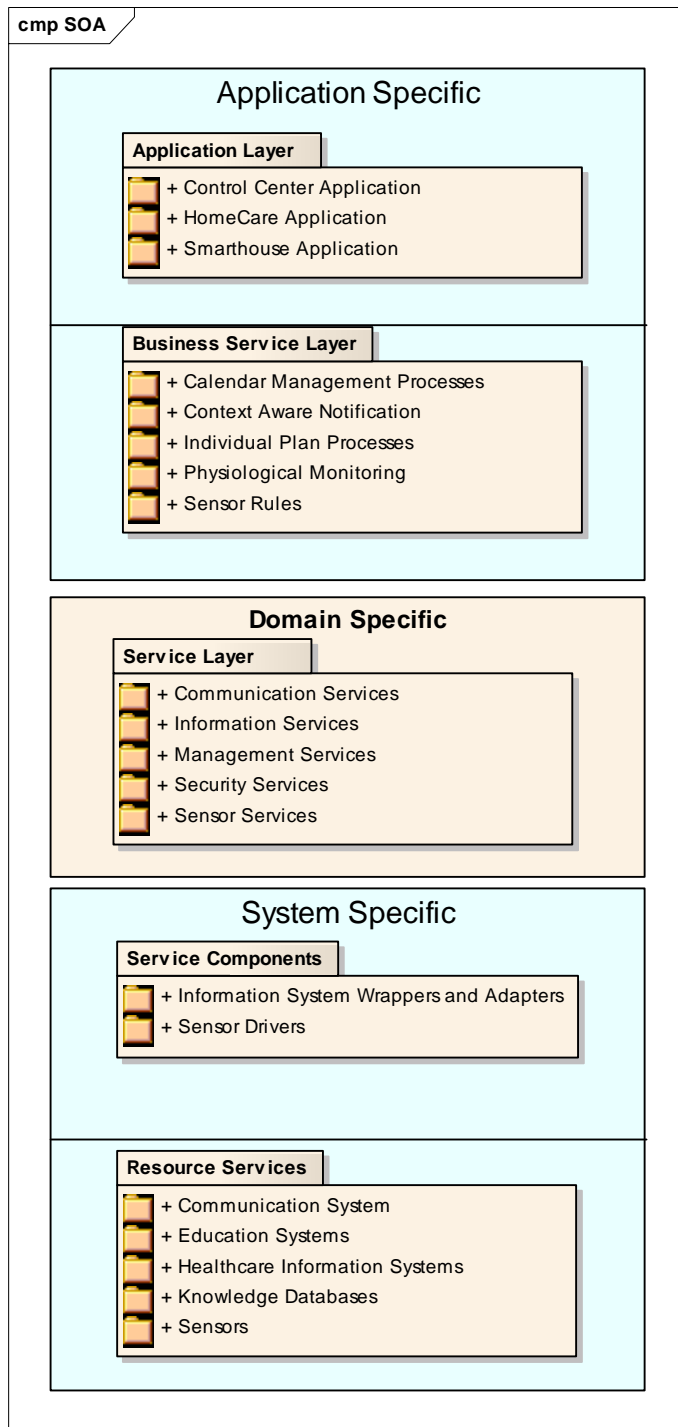


Figure 7: The MPOWER SOA Reference Architecture

The services in the Service Layer can be distributed and reused in a network, and accessed through standard interfaces over standard protocols.

6.3 MPOWER Information Models

The MPOWER information middleware is divided into three different components:

- Medical Information Model
- Social Information Model

- Context Information Model.

These models provide information about patient medical condition (medical information model), the patient's social network and social activities (social information model) and the current situation (context information models).

The services that use these information models should provide advanced homecare for the patient. The medical and social information models extend HL7v3 information models to support the needs of patient and professional caregivers. HL7 will be used as a basis to further develop models in the area of advanced homecare.

Information models outside the medical area that include information that is important for the social life of cognitive disabled and elderly people will also be developed and aligned with the ongoing work within the HL7 organization. The context information model will be developed in order to provide adoption of GUI dependent of the user needs based on the following parameters: application state, user type, location, terminal and mode of operation.

The overall process of defining information models that are relevant for implementing advanced homecare services is presented in MPOWER D3.1. It is worth mentioning that this process uses a “top-down” approach which means that it starts from detailed business requirements and process definitions and refines them in a stepwise fashion down to a software implementation. Business process models provide a blueprint for the identification of services.

6.3.1 Process of defining information models

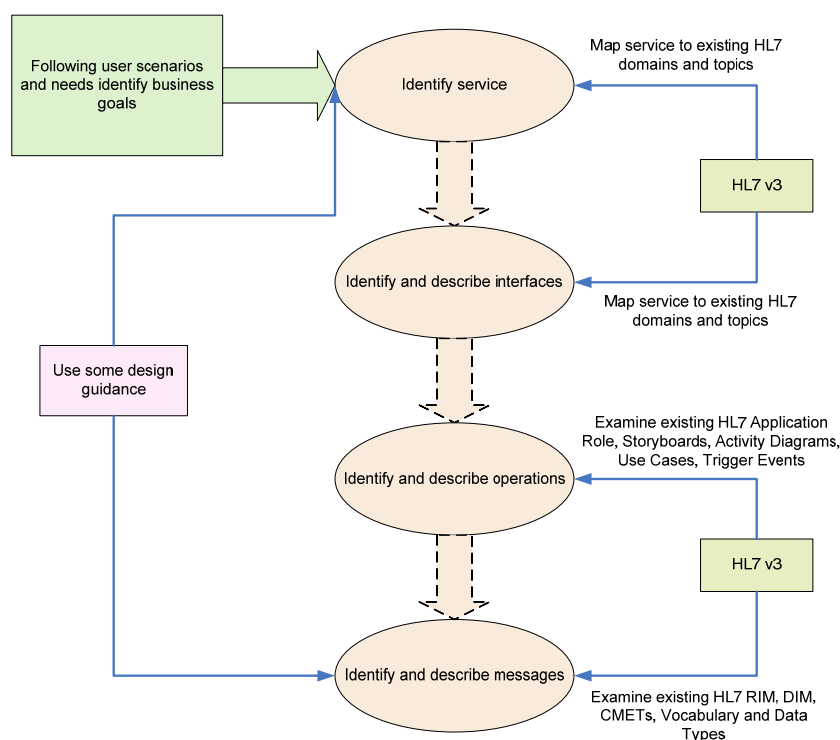


Figure 8 Process of defining information models

The overall process consists of few steps :

1. Start with user needs and try to think of the business goals you want to achieve (e.g. You want to provide subject-of-care with individual plan application)

2. Try to identify the services that will be choreographed in a new business process to fulfil the business goal (e.g. Subject-of-care needs medication list, calendar with all medication and non-medication activities, information's about his diagnose, level of dementia and overall medical condition)
3. Try to map these services to existing HL7 domains and topics (e.g. Scheduling Domain → Appointment Topic)
4. Start to identify the message content. At the business level this should be an information level in e.g. UML. This UML model is Platform Independent Model (PIM). At the Platform Specific Model (PSM)/Implementation level an XML Schema should be produced.

Using HL7 artefact following procedures should be taken out as part of this phase:

- If there is a matching CIM/R-MIM for the scope of the operation then this should be used. Otherwise you should start with DIM and identify appropriate classes in the scope.
- Look for relevant reusable information structures (CMETs)
- Data Types and Vocabulary should be based on existing V3 artefacts where they exist

7 MPOWER Middleware Services

The MPOWER middleware consists of a set of services and components that can be used by the application developers to rapidly design and implement an application. To enable reuse, the reusable services and components are provided in two different formats: as a UML Model or as a compiled component (with source code) ready to be deployed (see section 8.1 for details.)

The services provided as a part of the MPOWER Middleware are grouped into five categories as described below and illustrated in Figure 9: Management Services, Information Services, Sensor Services, Security Services and Communication Services. Details about each package of services are described in the following, and a general discussion on the usefulness of reusable service definitions is given in Appendix D.

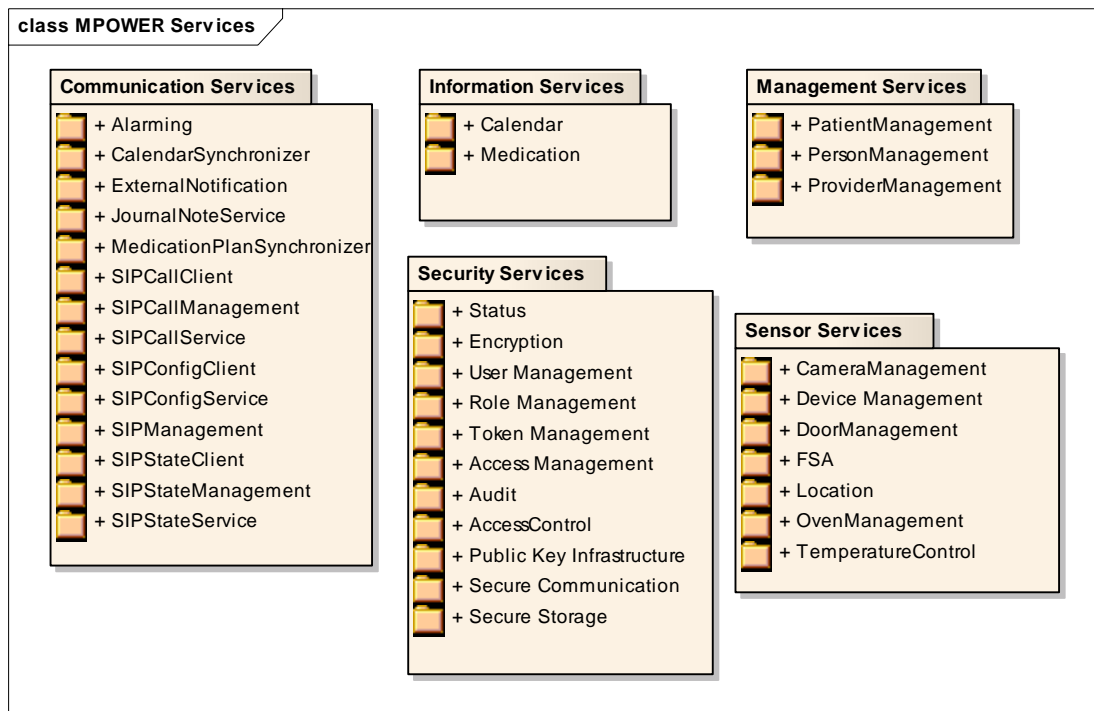
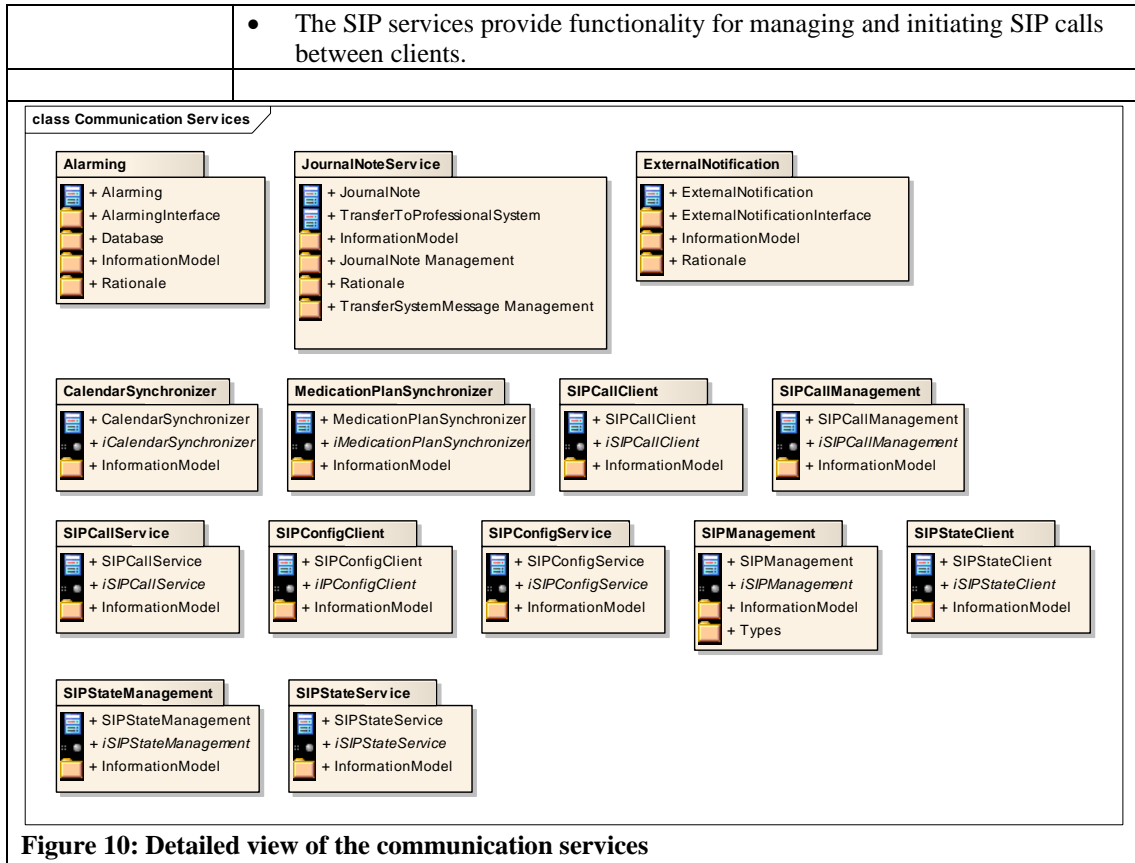


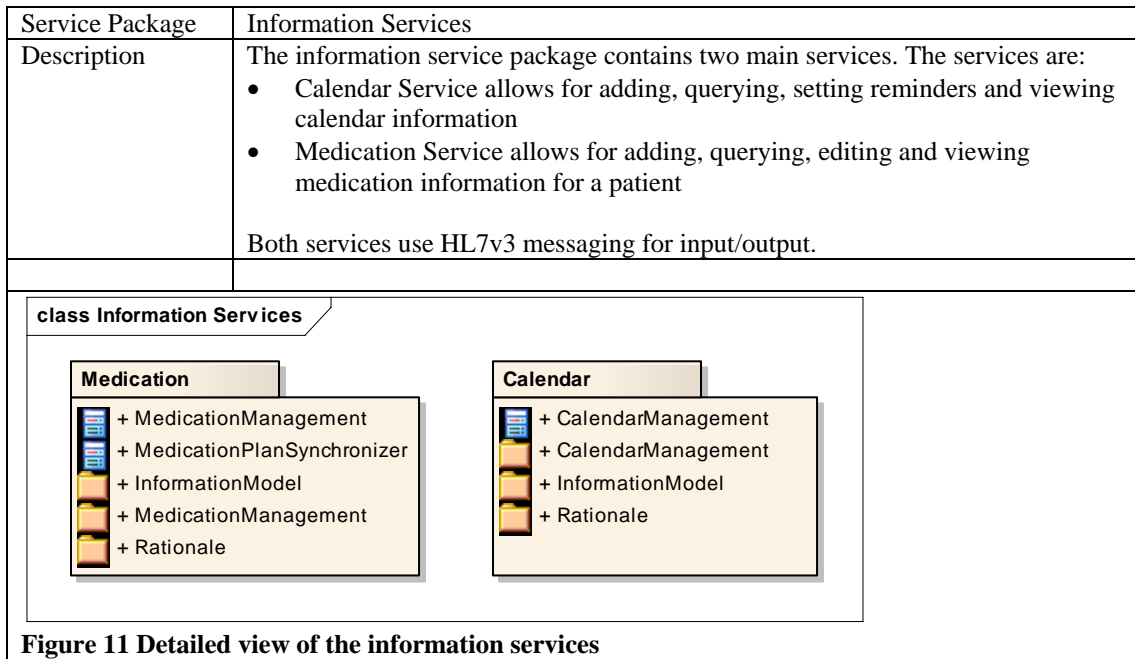
Figure 9: MPOWER Service Categories

7.1 Communication Services - (Logical diagram)

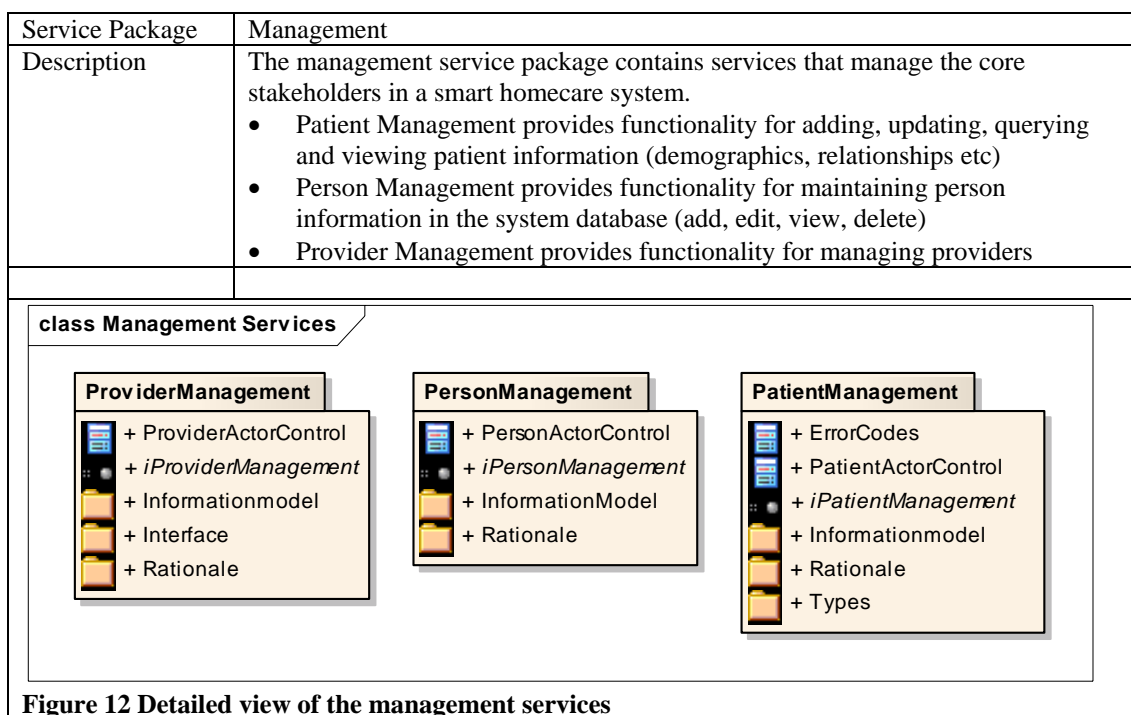
Service Package	Communication
Description	<p>The communication services package has services that communication information of different types. The package contains seven core services, in which the SIP communication service has several associated services.</p> <ul style="list-style-type: none"> • Alarming service provides functionality for managing alarms (trigger, stop, log, etc). it can be used by other services or applications • The Journal Note service enables transfer of local journal notes to an external system through a standardized interface. • External Notification provides operations for sending notifications in form o email, sms or messages to a specified recipient. • The Calendar Synchronizer service provides functionality for synchronizing two or more calendars. • The Medication Plan Synchronizer service provides functionality for synchronizing two or more medication plans.



7.2 Information Services - (Logical diagram)



7.3 Management Services - (Logical diagram)



7.4 Security Services - (Logical diagram)

Service Package	Security
Description	<p>The Security services package contains service that provides the well-known security mechanisms for authentication, access control and encryption. In addition, is has services for user management and auditing. The services are:</p> <ul style="list-style-type: none"> • User Management service enables the Administrator to manage the users of the system. The Administrator may add or delete users, update the user's role and get information about users • Token Management service is used by the Authentication and Authorization services to manage the login sessions, including issuing and controlling the validity of security tokens. • Audit service provides an interface enabling other services to store data that needs to be logged for future audit purposes. Additionally, it provides operations to retrieve logged data that is necessary for auditing. • Access management service manages the permissions and access profiles associated with the access control system. • Access Control includes the Authentication and Authorization services. The Authentication service verifies a user's credentials and allows access to the system only to users with valid credentials. The Authorization service determines what operations and which data an authenticated user can access, allowing access to resources only to legitimate, authorized users. • PKI service provides the interface for general management of certificates, i.e. issue, renew and revoke, and for verification of the validity of a certificate. • Role Management service enables the Administrator to manage the roles of the system. The Administrator may add or delete roles, assign users to roles, get information about roles, and view user's assigned to a role • Encryption mechanism describes functionality related to confidentiality and integrity protection of data. • Secure Storage mechanism describes functionality for storing data securely

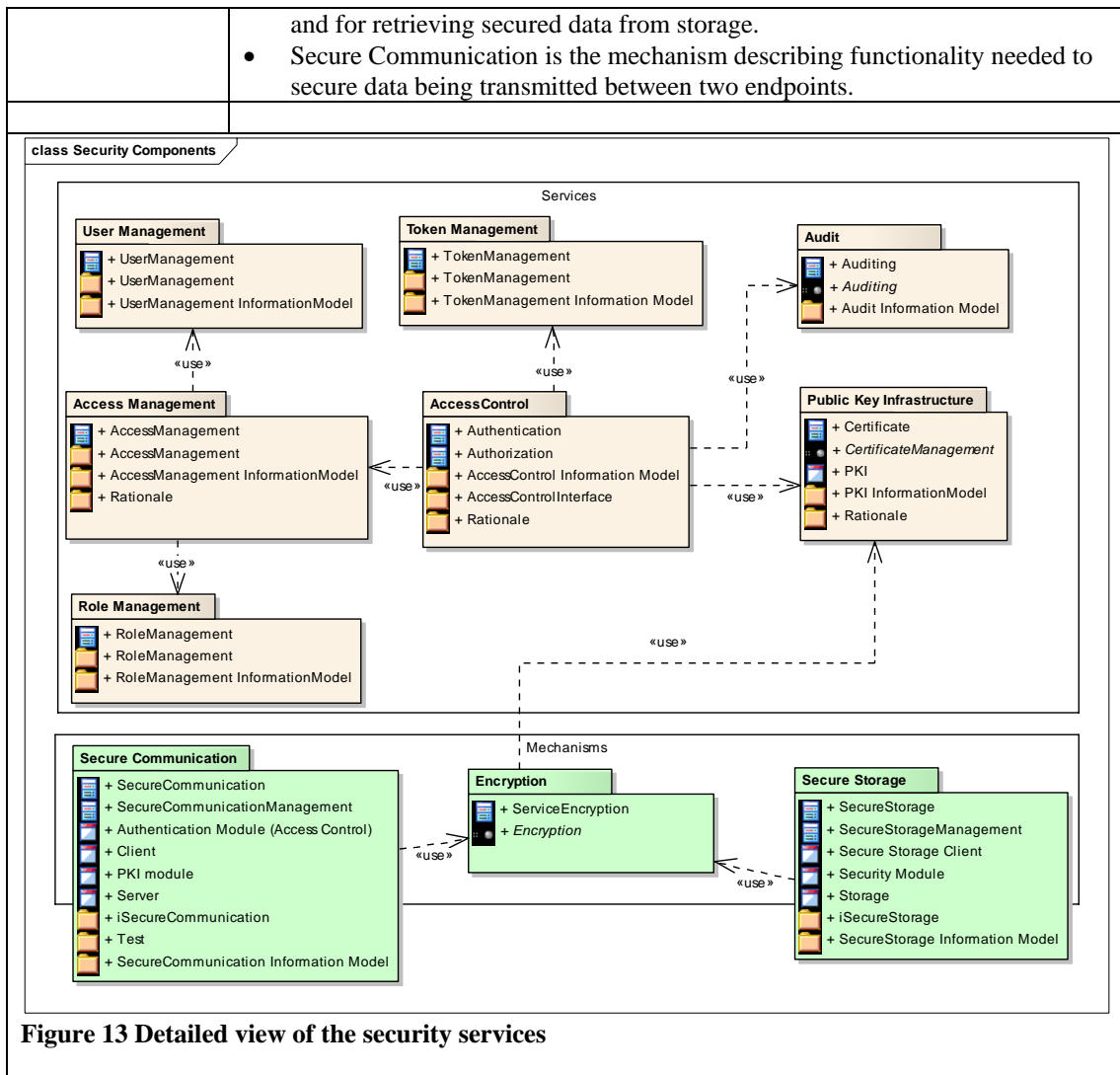


Figure 13 Detailed view of the security services

7.5 Sensor Services - (Logical diagram)

Service Package	Sensor
Description	<p>The sensor service package contains services the interface and manage sensors. The services expose both management mechanism and data access through an easy to use and standardized interface. The services are:</p> <ul style="list-style-type: none"> Device Management - this service provides necessary functionality to manage devices to be installed in the MPOWER environment. Possible operations are: adding new device information, removing a device or configuring an installed device. Temperature Control - a service providing a functionality of getting the measured values by the temperature sensors. Location - service through which the patient's location can be controlled at all time. Door Management - a service providing a functionality of getting the information from door magnetic contacts sensors. Camera Management - is a service through which an authorized access to MPOWER environment cameras is possible, with security features applied. FSA - is a component that serves to interoperate with different kind of sensors/devices in a unified way. It composes of layers where a data is

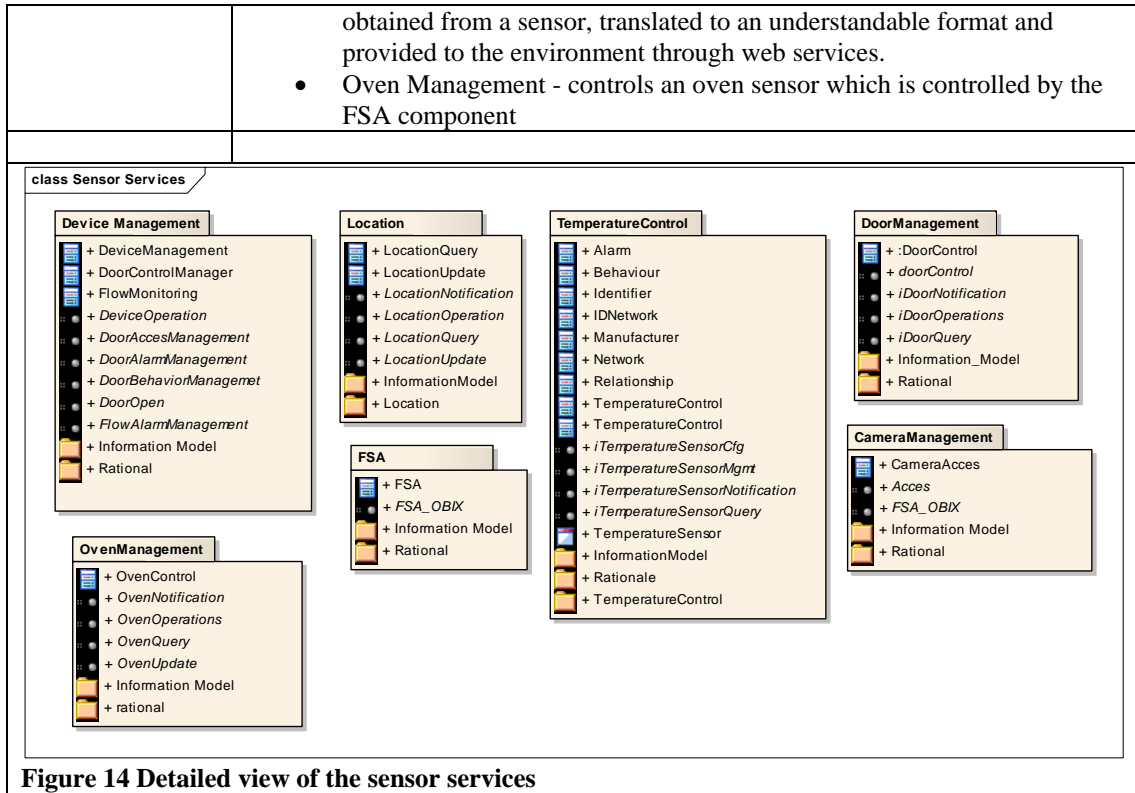


Figure 14 Detailed view of the sensor services

8 MPOWER Methodology

To address the information system interoperability problems, new techniques and methodologies have been introduced in the Software Engineering community. One of these is the Model Driven Development, or more precisely Model Driven Software Development (MDSO). The goals of MDSO described in [8] can be summarized as follows:

- Increase development speed and software quality through automation
- Higher level of reusability as the architectures, modelling languages and transformations are generic for the domain (abstract)
- Improved manageability of complexity through abstraction
- MDSO is based on the Object Management Group's Model Driven Architecture ® (MDA). OMG's focus is on interoperability, portability and reusability through architectural separation of concerns [9]

8.1 Model-Driven Software Development

In 2002, the Object Management Group (OMG) introduced the Model-Driven Architecture (MDA)[10], an approach focusing on using models (e.g., UML models[11]) as first-class entities in the development of software systems. In practice, this means that the models are used directly in the implementation of an information system, either as system blueprints or as input to code generation engines that produce executable code. *MDA is the most known model-driven software development (MDSO) approach, and the overall idea is to separate business functions (in Platform Independent Models - PIM) from its technological implementations (in Platform Specific Models – PSM), enabling code generation and reuse of components. The overall benefit is improved interoperability and reduced development time and cost.*

The model-driven software development approach (MDSO) [8] has been, and still are, subject for debate in the software engineering community. Despite passionate criticism, many organizations have reported significant improvements in the development process and on software quality [12, 13]. In the healthcare domain, using Model-Driven Architecture[10, 14] has been presented in several articles [15-20].

Several initiatives are working on solutions where many of the resource consuming integration tasks are automated, by providing ready to use modules or using a code generation engine to produce executable code. Service Oriented Architecture (SOA)[1, 2] is currently the best practice approach to enable applications to expose information and services towards each other. Kawamoto and Lobach [19] argue that the properties of SOA provide simpler software design, improves software reusability and save development cost.

To design SOA services, a formal language such as UML should be used to ensure correct and intentional specifications. A MDA-compliant tool with an UML extension such as the IBM UML Profile for Software Services [21] provides a formal framework from which executable SOA services can be generated.

Using an MDSO approach in the development of healthcare information system services could facilitate the use of standards through specification of reusable standards-based PIMs. Advanced UML mechanisms such as Profiles and Patterns could be used to further extend the expressiveness of the modelling language and force the use of standardized healthcare concepts. As a result, the developed systems will increase the level of interoperability, and at the same time development and maintenance costs will decrease.

MDSO seeks to use models (a formal graphical notation) to represent all artefacts involved in the development of a software system. Models are both abstract and formal at the same time, meaning that irrelevant details are abstracted away and the core is described (modelled) unambiguously. The models are used in diagrams that specifies a static or dynamic (behaviour) view of the target system. Diagrams are typically created using a top-down approach where the high-level concepts are identified and documented before they are broken down into sub-concepts, workflows and information models. The low-level detailed models can be transformed into new and technology specific models (including concepts from J2EE or .Net). From the technology specific models, executable code can be generated.

8.1.1 *OMG's Model Driven Architecture (MDA)*

The best known MDSO approach is OMG's Model Driven Architecture (MDA)[10, 14]. MDA provides an open, vendor-neutral approach to the challenge of business and technology change. Based on OMG's established standards, the MDA separates business and application logic from underlying platform technology. The platform-independent models (PIM) of an application or integrated system's business functionality and behaviour, built using UML and the other associated OMG modelling standards, can be realized through MDA on virtually any platform, open or proprietary, including Web Services, .Net, CORBA, J2EE, and others. These platform-independent models document the business functionality and behaviour of an application separate from the technology-specific code that implements it, enabling interoperability both within and across platform boundaries. No longer tied to each other, the business and technical aspects of an application or integrated system can each evolve at its own pace – business logic responding to business need, and technology taking advantage of new developments – as the business requires[9].

8.1.2 *Model Transformation and Code Generation*

One of the core features of MDSO (and MDA) is the ability to transform one model of the system into a new technology specific model which in turn can be used to generate executable code. The overall concept is to model the system from different viewpoints, each viewpoint having its own goal and role in the development process. MDSO is in line with MDA that describes three different viewpoints and their corresponding models, namely the Computation Independent Model (CIM), Platform² Independent Model (PIM) and Platform Specific Model (PSM)[9].

- CIM: The CIM is a model that focuses on the environment of the system, and the requirements for the system; the details of the structure and processing of the system are hidden or as yet undetermined.
- PIM: The platform independent viewpoint focuses on the operation of a system while hiding the details necessary for a particular platform. A platform independent view shows that part of the complete specification that does not change from one platform to another.
- PSM: The platform specific viewpoint combines the platform independent viewpoint with an additional focus on the detail of the use of a specific platform by a system.

² A *Platform* in MDA is defined as a set of subsystems and technologies that provide a coherent set of functionality through interfaces and specified usage patterns, which any application supported by that platform can use without concern for the details of how the functionality provided by the platform is implemented

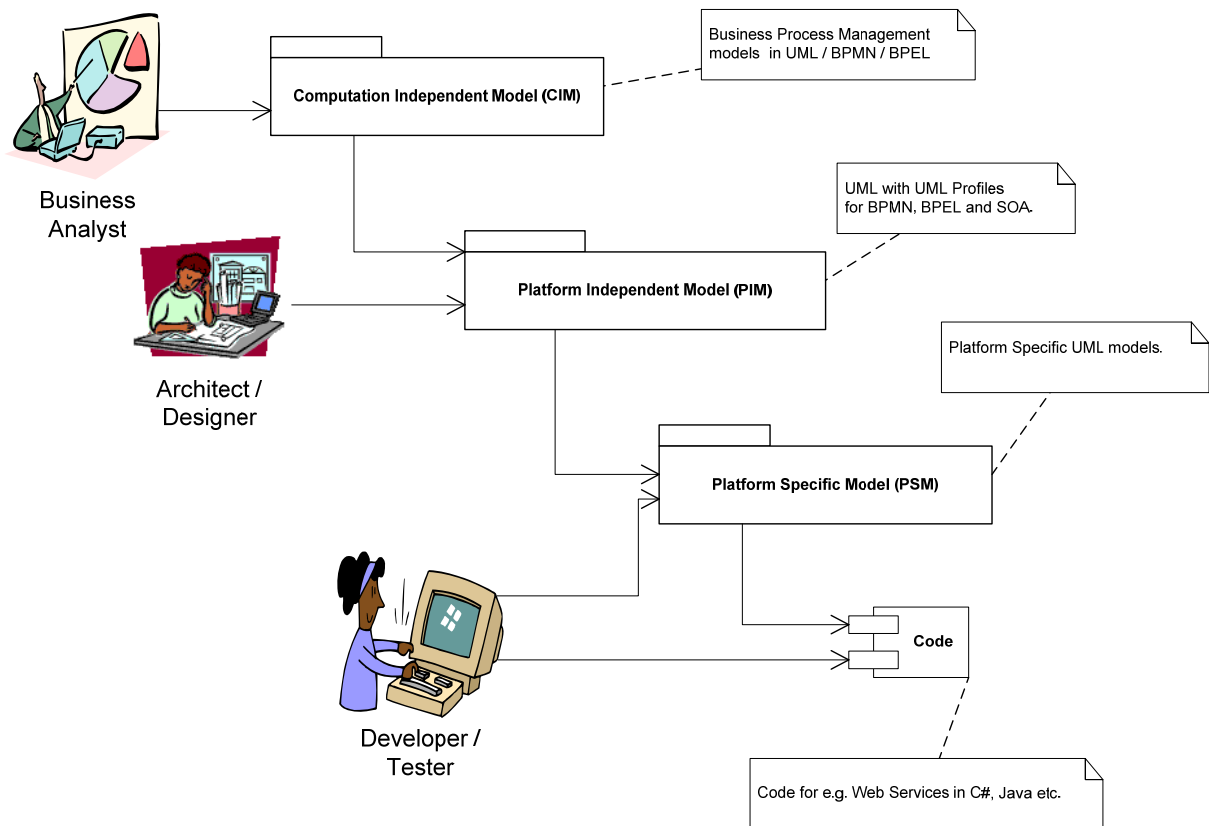


Figure 15 : The MDA models. Figure based on [22]

8.1.3 Meta-models, UML Profiles and UML Patterns

One way to achieve semantic and syntactical information interoperability is to have a common meta-model. A meta-model (data about data) describes the semantics of a language, such as an information standard in healthcare, and must be described formally. In a model-driven development project one may use several meta-models, but to ensure interoperability one should have a mapping model between the different meta-models. A meta-model may be domain specific, e.g. a meta-model for continuity of care, providing a familiar language to the people modelling information systems for this domain.

OMG has specified the Unified Modeling Language (UML) [11], a formal modelling language that can be used to specify (model) both static and behavioural aspects of a system. Using the Object Constraint Language (OCL)[23, 24] together with UML enables the modeller to enrich the models with enough detail to render it possible to generate high-quality source code. Another mechanism offered by UML is the use of UML Profiles. A profile is used to add domain specific concepts and terms to the modelling language. Stereotypes, tagged values that can be applied to elements, attributes, associations etc, enable accurate modelling for a specific domain.

A Design Pattern as described by Gamma et al [25] is a powerful mechanism for specifying reusable and well-proven concepts. UML can be used to describe a pattern formally, and most UML modelling tools support the use of UML Patterns in system modelling.

8.2 SOA, MDSD and HealthCare - a way to improve the systems' compliance to standards

8.2.1 Conceptual Model

The core of MDSD is to use formal models (e.g. UML models) to express structure of behaviour of a software system and then use the models as basis for code generation. The models must include enough information (richness) to the transformation process so that as much of the software system code as possible can be generated. To aid the software modeller in adding information, UML Profiles can be used. A UML Profile includes stereotypes, tagged values and constraints that can be assigned to a UML modelling element during design. The transformation engine will use this information to generate (more) complete code.

The process of creating a standards-based Healthcare Application based on MDSD and reusable services can be summarized in three steps:

- 1) Create UML Profile and Model Transformation from Healthcare Standard
- 2) Create reusable Healthcare Middleware Service applying UML Healthcare Profile
- 3) Create Healthcare Application using UML Healthcare Profile and reusing Healthcare Middleware Service(s)

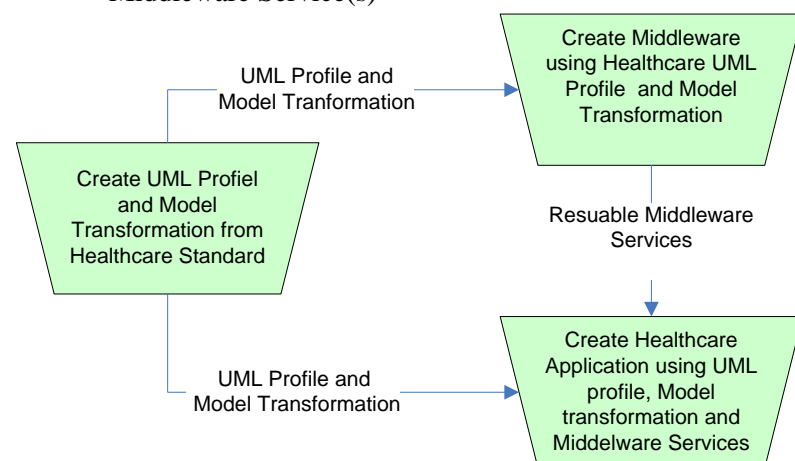


Figure 16: High-level conceptual process model

8.2.2 Create UML Profile and Model Transformation from Healthcare Standard

A UML Profile provides domain specific information that can be used during system design and modelling. The profile can hold UML extensions in the form of stereotypes, tagged values and constraints. Using a Healthcare Information Standard such as CONTSYS[26], stereotypes such as “HealthcareProfessional” and “SubjectOfCare” can be specified. Tagged values can be a Boolean value for “isOrganDonor” and “BloodType”. An example of a constraint can be that a “SubjectOfCare” must have a “BloodType”.

A Model Transformation is a “script” that uses the UML models as input to produce another more specific model or text (code). The Healthcare Model transformation described here will use the extensions specified in the UML Profile as input to the transformation process. E.g., a UML Class extension called “SubjectOfCare” can make the transformation engine produce a class that has operations for setting and getting the “BloodType” (TaggedValue).

The figure below shows the steps involved.

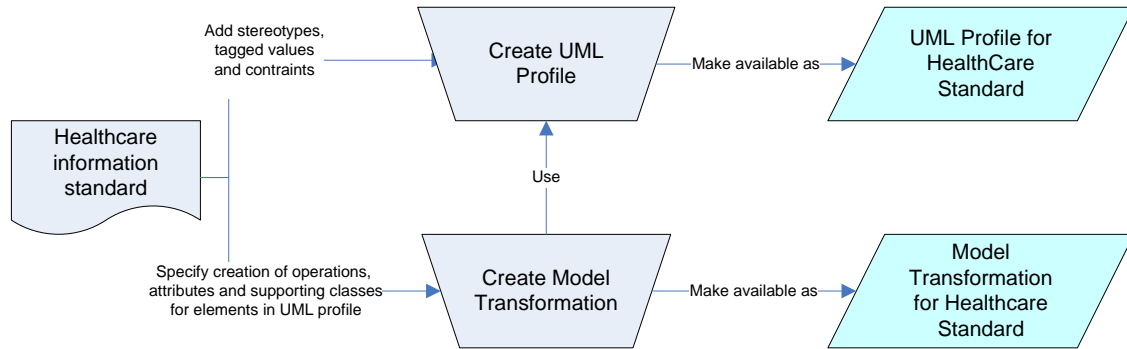


Figure 17: Creating a UML Healthcare profile and Model transformation to support a Healthcare Information Standard

8.2.3 *Create Healthcare Middleware Service using UML Healthcare Profile*

Using the Healthcare UML Profile and Model Transformation, healthcare specific middleware services can be created.

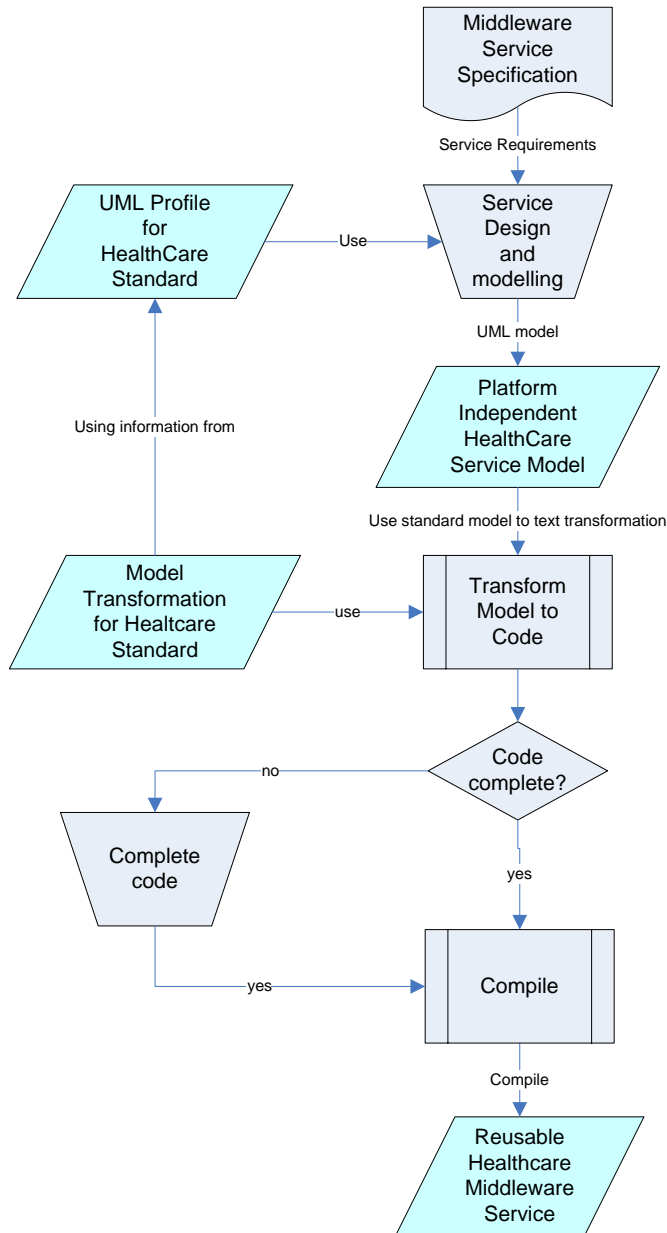


Figure 18: Creating reusable Healthcare Middleware Services using Healthcare UML Profile and Model Transformations

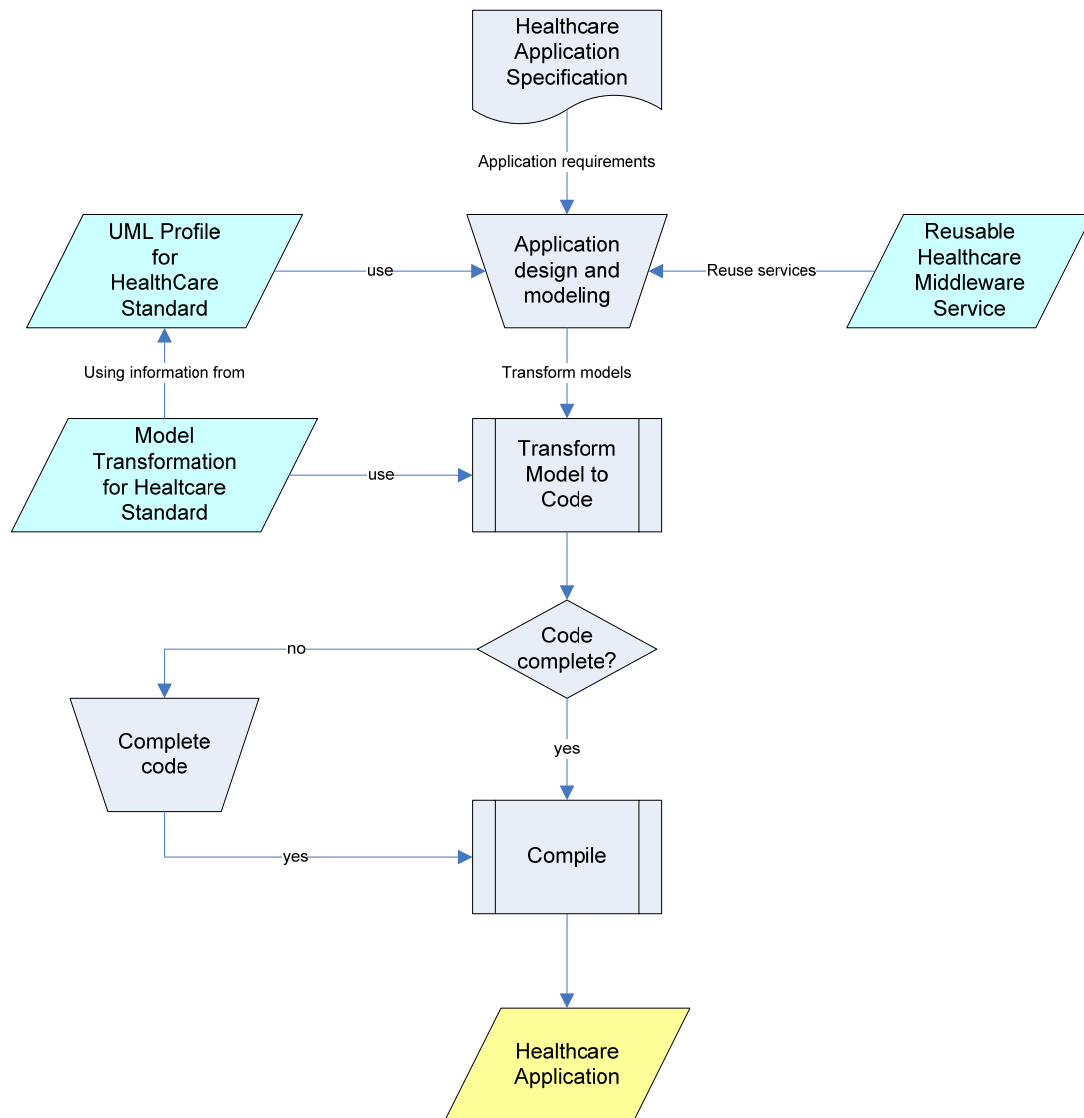


Figure 19: Creating a Healthcare Application using Healthcare UML Profile, and Model Transformation and reusing Middleware Service(s)

8.3 MDA Tool Support for Specifying Services in MPOWER

The development of the MPOWER middleware platform follows the model-driven approach as defined by the Model Driven Architecture from OMG [10]. To document the domain business activities, the user needs, user scenarios, formal usecases, and system features relevant for MPOWER are modelled in UML. Understanding of user needs and user involvement is central to the MPOWER approach. This knowledge must be used in the development process. The model-driven development approach seeks to represent all artefacts involved in the development process as model elements; hence activity scenarios for the user needs should be modelled in UML as UseCases and activity diagrams.

The services are modelled in UML as class diagrams, and using stereotypes from the UML profile for System Services [21]. Mikalsen et al demonstrates the use of this profile for interoperable healthcare services in [27]. The MPOWER services are modelled following the design principles by Erl [1] and SOA4HL7 [7]:

- Identify a service: As stated in SOA4HL7 document, it is very difficult to provide a precise answer to the question “What makes a good service?” However, the interfaces and operations included should all be closely related in a business sense and part of the same overall function with similar purpose and be fairly complete for that function.
- Identify interfaces: Service definitions can include one or more interfaces. In general, different interaction styles can be split into different interfaces e.g. Query (read-only) vs Update vs Notification (subscription based).
- Identify operations: The operation is the actual “unit of functionality” that needs to be carried out. One interface can contain many operations.
- Identify message content: Each operation produces and consumes specific message content. For the operations at the PIM level this can be represented as UML model. At the implementation level, assuming a web service solution, an XML Schema should be produced.

8.3.1 Information Modelling – HL7 standard messages

One of the key features of the MPOWER middleware services is to make it easier for system developers to adhere to international healthcare information standards. HL7 version 3 messages are the primary standard to use for message content in MPOWER, the last phase of the service specification methodology. HL7 Version 3 standard use a well-defined methodology based on a Reference Information Model (RIM) [28]. The RIM represents the essential part of the HL7v3 development methodology, as it provides an explicit representation of the semantic and lexical connections that exist between the information contained in HL7 messages. The methodology for using HL7 in SOA development is described in [7].

As specified in the MDA Guideline document [10], the PIMs are transformed into models that incorporate implementation details specific for a certain technology, to form the Platform Specific Models. The target platform of the MPOWER SOA services were decided to be Web Services [29] developed in Java EE 5 [30]. The main PSM artefact in web service specification is the description of the web service in a WSDL file[31]. A WSDL file contains details about the data types, messages used, operations provided, protocols used, and the location of services(s). From a WSDL file, much of the needed implementation code can be generated using appropriate tools.

8.3.2 Modelling: Computation Independent Models - User needs

The process of capturing the user needs and business aspects (the CIM) resulted in 18 problem and activity scenarios, each with two to four activities (See deliverable MOWER D7.1 for details). Each scenario is described by one or more UML use cases and grouped into 10 logical packages. Each use case is related to a set of actors and contains a reference to the scenarios from which it is derived. Figure 20 shows the “Stakeholder management” use case and how it is related to actors and other use cases.

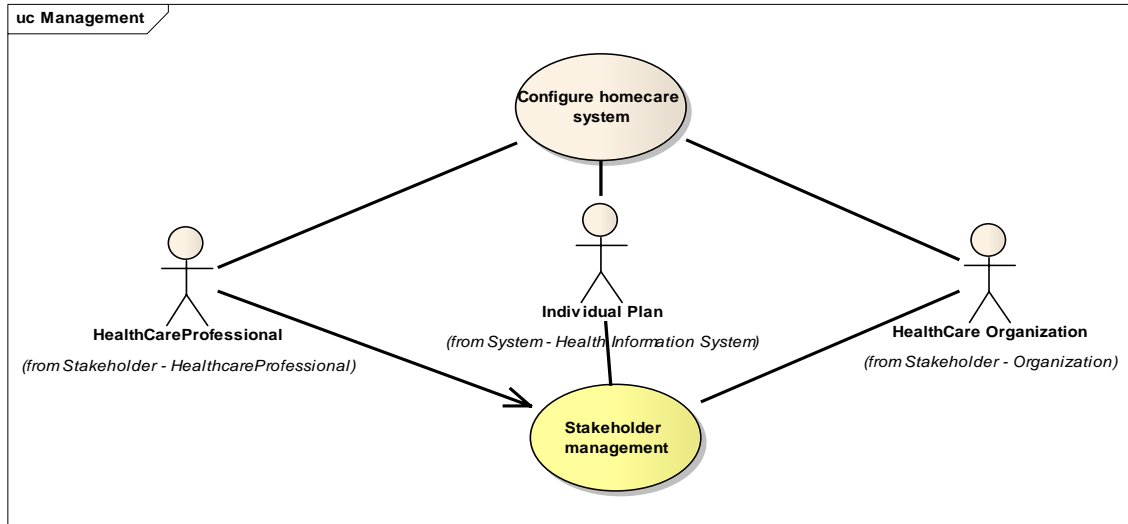


Figure 20: Use Cases for Management scenarios.
The traceability from *use case* to the *scenarios* is shown in Figure 21.

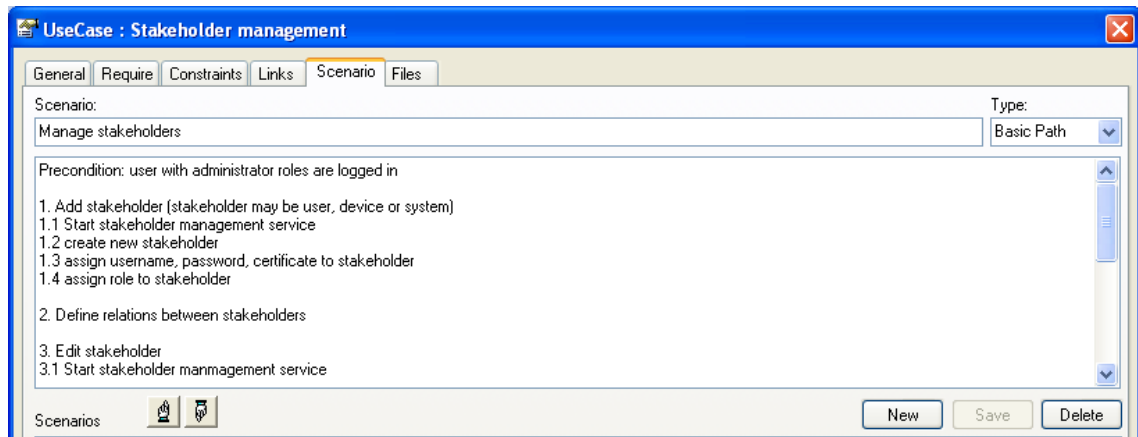


Figure 21: Use Case "Stakeholder management" and the related scenarios

From the MPOWER *use cases* and *scenarios* a total of 168 *features* are described and grouped into 14 categories. Each *feature* is directly related to one or more *use cases* as shown in Figure 22 where the Stakeholder management *use case* is related to four *features*.

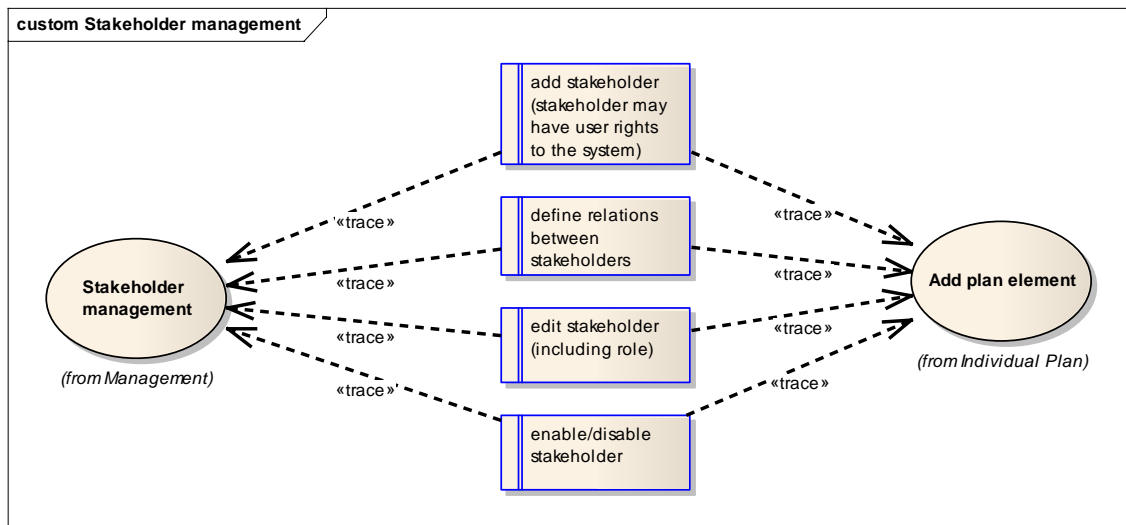
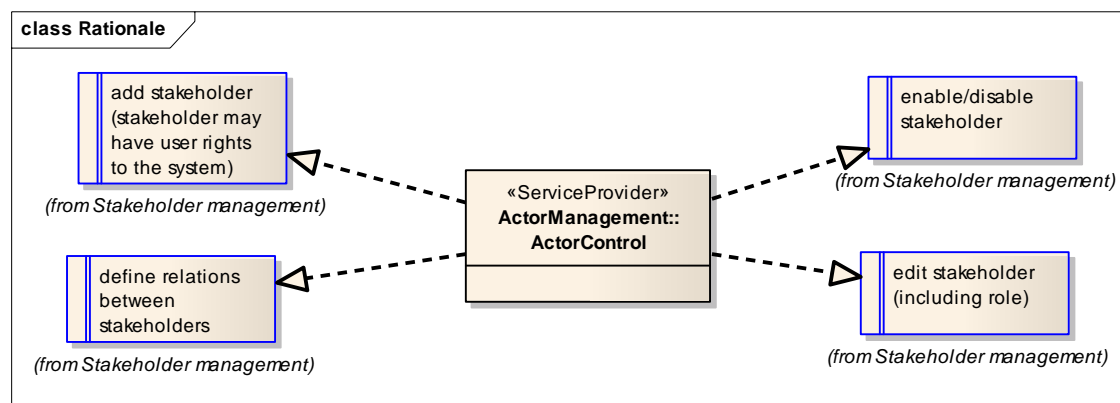


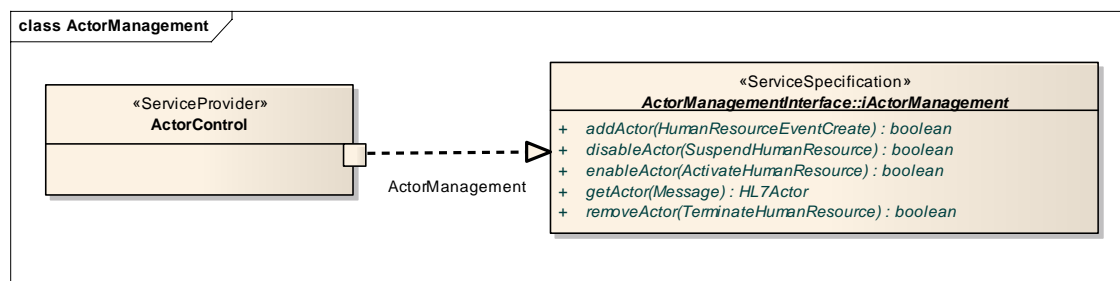
Figure 22: Features derived from Stakeholder management and Add plan element use cases

8.3.3 Modelling: Platform Independent Models - Service Specification

As presented in section 7, five packages of MPOWER middleware services are specified from the features: management, communication, sensor, security and information services. The rationale for the services is explicitly described as UML implement dependencies as shown in Figure 23.

**Figure 23: Service rationale. The MedicationManagement service implements five features**

Each service package contains a number of atomic services that are specified using UML with the IBM SOA UML Profile [21]. Figure 24 shows the ActorManagement ServiceProvider (UML Class) from the Management Services package providing one service (UML Port) with one service specification (the UML Interface iActorManagement). The operations on the interface reflects the features that the service implements.

**Figure 24: The Service Model with UML Stereotypes**

The messages used in the operations on the interface are specified according to HL7. Existing messages provided from HL7 are used when appropriate whereas new messages are developed using the tools provided by HL7. For the ActorManagement Service used as example, all messages involved are specified in the HL7 Personnel Management domain models³. The HL7 message definitions are available for download as XML Schema files.

8.3.4 Modelling: Platform Specific Models - WSDL Models and Code

The MPOWER middleware services are specified to be implemented as web services. Web services are described using the Web Service Description Language (WSDL) and the services modelled as PIMs are automatically transformed into WSDL models using a transformation script. In the context of MDA, the WSDL model is regarded as PSM. The WSDL model is used to generate WSDL code.

³ HL7 Personnel Management domain: <http://www.hl7.org/v3ballot/html/domains/uvpm/uvpm.htm>

8.3.5 The MPOWER Tool Chain

A tool chain to be used in MPOWER must support development of web services following the method described in the previous. In addition, the toolchain must provide functionality for implementation, deployment and testing.

The MPOWER Toolchain functionality is presented by applying the toolchain in a MDA process for developing homecare services. The CIM shows how knowledge is incorporated in the domain analysis phase and used for PIM specification and PSM generation. The models are presented with examples from the “ActorManagement” functionality. The MPOWER toolchain is described in the last part of this section.

To support the model-driven software development process described in the previous, three main process activities must be supported:

- UML Modelling, model transformation and code generation: Enterprise Architect (EA) from Sparx Systems was chosen as the primary UML modelling tool. EA supports the latest version of UML and XMI [32] that allows for sharing of models between tools. EA also provides an easy to use model transformation and code generation engine that is customisable. The core UML model in EA can be extended using UML Profiles. EA does not require a powerful PC and is quite inexpensive compared to tools with similar functionality
- Information Modelling: The tools recommended by HL7 were used for HL7 message modelling[28].
- Integrated Development Environment and Application Server: Java EE 5 bundle with Netbeans 6.0/6.1 and Java Sun Application Server 9⁴ was chosen for web service implementation, deployment and hosting. For testing web services using SOAP transport, the SOAPUI NetBeans plugin⁵ can be used in addition to the built-in Netbeans Web Service testing feature.

Figure 25 shows the tools, activities and artefacts involved the engineering process. The MPOWER UML Service model is created in EA based on the User Scenarios and Needs document. HL7 messages are related to the service model and integrated during WSDL generation. The WSDL file is imported into NetBeans and a web service is generated along with a SOAPUI test project. Using the Netbeans IDE a web service is fully implemented and deployed to the Sun Java Application Server. The SOAPUI plug-in for NetBeans provide a powerful and easy-to-use test mechanism for web services.

⁴ NetBeans website: <http://www.netbeans.org>

⁵ SOAPUI: <http://www.soapui.org>

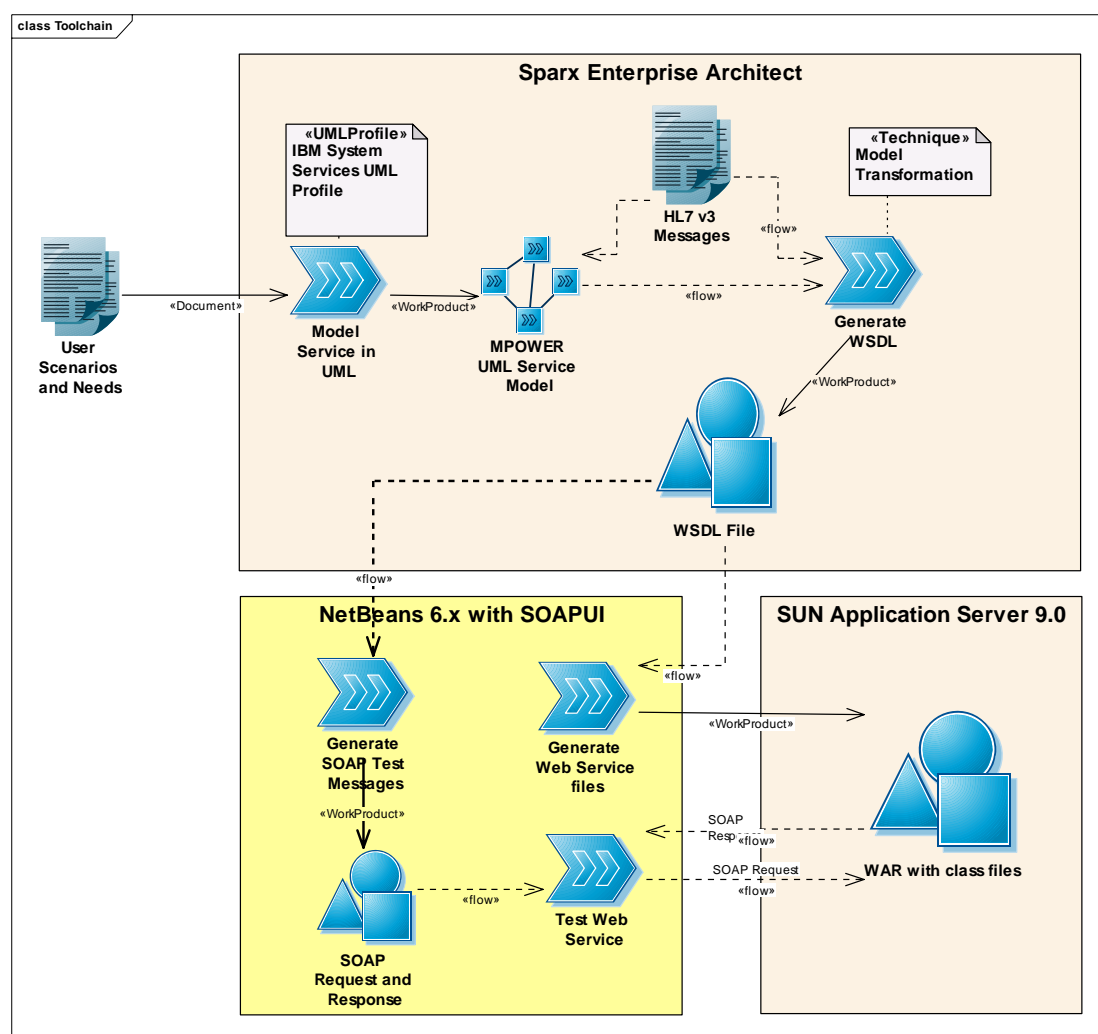


Figure 25: The MPOWER Tool Chain and Artefacts

8.3.6 Tool Chain Example

To demonstrate the toolchain and development process, an example from the MPOWER middleware development is presented. The example is the ActorManagement service shown in the previous.

1. Service identification: The service is identified through an analysis of features and use cases as shown in Figure 20 and Figure 22.
2. Interface identification and operation specification: the features provide important information about the type of operations to be grouped into an interface. The use cases with scenario descriptions provide complete behaviour (operation) specification that is required on the interface (Figure 23, 25 and 26)
3. Message Content identification: The Personnel Management domain message types were appropriate for use on the ActorManagement interface. XML Schemas for the message types are available for download from the HL7 website [33].
4. Service Modelling: The service was modelled in UML extended with the IBM UML Profile for Software Services [21]. The documentation of the UML Profile provides guidelines for use.
5. WSDL model transformation and WSDL code generation: Enterprise Architect provides a transformation to WSDL models. The default transformation had to be customised with additional namespace and protocol information. Figure 26 shows the WSDL model for the ActorManagement service.

6. WSDL import and Web Service creation: Netbeans 6.0/6.1 has a simple interface for creation of web service from WSDL files. All necessary classes with attributes and operations are generated.
7. Implementation, deployment and testing: NetBeans offers a standard integrated development environment (IDE) where the service's detailed functionality can be implemented before it can be deployed to the bundled Java Sun Application Server. SOAPUI is integrated into the NetBeans IDE and enables testing and debugging of SOAP service interactions.

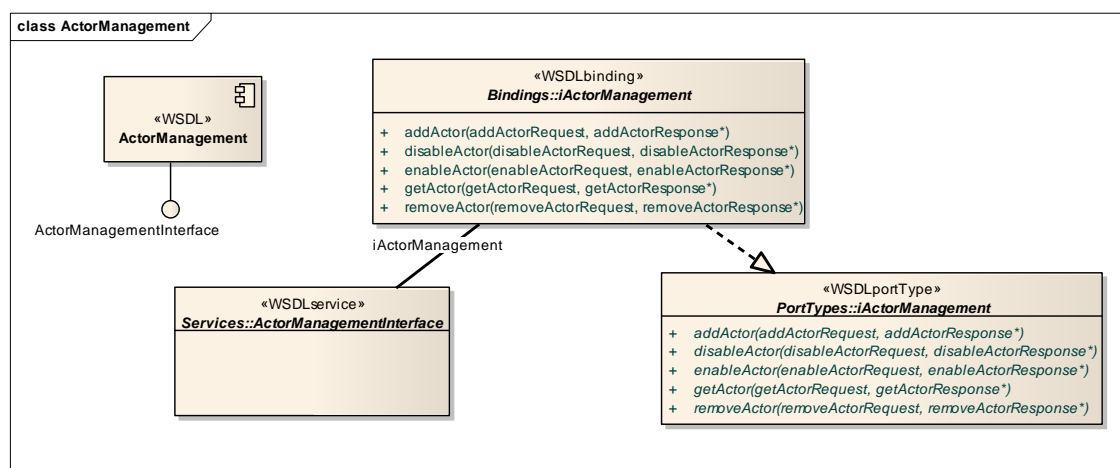


Figure 26: WSDL model for ActorManagement Service

8.4 Where MDA should be used in MPOWER

MPOWER will apply an MDA approach in:

1. Definition of the domain: This will result in the Computation Independent Models (CIM) of the MPOWER (core) middleware. The identified user needs and workflows will be documented as UML Usecases, with references (for traceability) back to the originating user scenario.
2. Specification of middleware services: The Platform Independent Models (PIM) of the MPOWER middleware. Following the guidelines for service modelling in UML given by IBM [7], all services will be modelled with service names, operations names, message names and message types.
3. Implementation of middleware services: Both Platform Specific Models (PSM) and code for the MPOWER middleware. The services modelled in PIM will be transformed into PSM using an adapted transformation script.
4. Implementation of applications (proof of concept): CIM, PIM, PSM and code for the application. From the PSM models, code will be generated using code generation scripts.

The degree to which MDA will be used will vary. However, the following principles exist:

- Models should be the primary artefacts of documentation
- Developers should seek to use MDA mechanisms where appropriate
 - Differentiate between PIM and PSM
 - Use Model transformation to create PSM from PIM
 - Use Code generation from PSM to Code

Platform Artefact	Smart Home	Care Center	Common Service Provider
Application Server	<i>Sun Application Server 9.x / Glassfish v2</i>	<i>Sun Application Server 9.x / Glassfish v2</i>	<i>Sun Application Server 9.x / Glassfish v2</i>
Database	<i>Hibernate using MSSQL, MySql, Oracle</i>	<i>Hibernate on top of legacy system database layer</i>	<i>Hibernate</i>
Business Execution Server	<i>Optional. Open ESB v2 recommended</i>	<i>Open ESB v2</i>	<i>Optional</i>
Messaging Server	<i>Java Messaging Server</i>	<i>Java Messaging Server</i>	<i>Java Messaging Server</i>
Client Terminal	<i>Tablet PC</i>	<i>PC</i>	<i>Optional</i>
Server	<i>Laptop PC or Stationary PC</i>	<i>Server</i>	<i>Mainframe</i>
Network (minimum)	<i>ADSL/SDSL</i>	<i>SDSL</i>	<i>SDSL</i>

Table 2: Platform artefacts and implementation technologies

9.1 MPOWER Recommended Deployment Platform

MPOWER services are implemented as Web Services [29, 31] that can be composed into business process services using Business Process Executive Language (BPEL) [34].

For data persistence, Hibernate and any some alternatives of databases can be used. MySql and Oracle XE are two eminent and free alternatives. MPOWER uses Hibernate as a data access layer from the web services.

Open ESB v2 provides the required BPEL support a part of the BPEL Engine. A built-in XSLT engine in Open ESB v2 can be used for XML transformation at the business service layer

9.1.1 Application Server

The Java System Application 9.1 is based on [Project GlassFish](#), a community building an enterprise class Java EE 5 application server with clustering for scalability and availability, advanced administration and best-in-class performance. The Java System Application Server 9.1 is available with cost-efficient annual subscriptions. Customers no longer have to choose between open source and enterprise features.

[Java System Application Server Platform Edition 9.0 Update 1](#) is the industry's first free, robust, commercial-grade Java EE 5 compatible application server. Starts faster, uses less memory and incorporates Java EE 5 features such as EJB 3.0, JSF 1.2 and annotations that help developers write and deploy applications more quickly using 30-90% less code. The server represents the latest Java technology that makes building robust, scalable enterprise applications easier than ever and is the perfect platform for implementing SOA and Web 2.0 applications.

Glassfish V2⁶ is a free, open source application server which implements the newest features in the Java EE 5 platform. The Java EE 5 platform includes the latest versions of technologies such as such as JavaServer Pages (JSP) 2.1, JavaServer Faces(JSF) 1.2, Servlet 2.4, Enterprise JavaBeans 3.0, Java API for Web Services(JAX-WS) 2.0, Java Architecture for XML Binding(JAXB) 2.0, Web Services Metadata for the Java Platform 1.0, and many other new technologies.

9.1.2 Business Process Execution

Open ESB V2 Preview 3⁷: Open ESB V2 Preview 3 implements a Java Business Integration (JBI) runtime that incorporates the JSR 208 specification for Java Business Integration and other open standards. Open ESB allows you to integrate web services and enterprise applications as loosely coupled composite applications, realizing the benefits of a service-oriented architecture (SOA).

Open ESB supports pluggable service engines and communication protocol bindings as well as dynamic, configurable, message management and delivery. When installed with Java Application Platform SDK Update 3 Beta or NetBeans IDE 6.0 Beta 1, Open ESB includes the JBI Runtime and the service engines and binding components listed below. Developers can also create additional plug-in components to fit specific integration tasks.

Component/Feature	Description
JBI Framework	Runtime that implements a JBI instance. The JBI Runtime is a standard feature of Sun Java System Application Server 9.1.
BPEL Service Engine	Provides services for executing Web Services Business Process Execution Language 2.0 (WS-BPEL, or BPEL) compliant business processes.
Java EE Service Engine	Connects Java EE web services to JBI components.
SQL Service Engine	Provides SQL execution services to other JBI components.
XSLT Service Engine	Transforms XML documents using XSL style sheets.
File Binding Component	Provides a transport service to a file system and offers a comprehensive solution to interact with the file system from the JBI environment.
HTTP Binding Component	Provides external connectivity for SOAP over HTTP in a JBI 1.0 compliant environment.
JMS Binding Component	Provides Java Messaging Service (JMS) transport for inbound and outbound messages.

⁶ <http://www.sun.com/software/products/appsrvr/index.xml> <https://glassfish.dev.java.net/>

⁷ http://java.sun.com/integration/opensb2_0/

9.1.3 Databases and Data Access

MPOWER Services need to access data in different databases as well as store platform-related information such as context information and rules. Hibernate is chosen as the platform data access layer whereas several databases can be used as the underlying repository.

Below is a summary of the technologies. References to the original source, downloads and more details are given.

- **Hibernate:** Hibernate⁸ is a powerful, high performance object/relational persistence and query service. Hibernate lets you develop persistent classes following object-oriented idiom - including association, inheritance, polymorphism, composition, and collections. Hibernate allows you to express queries in its own portable SQL extension (HQL), as well as in native SQL, or with an object-oriented Criteria and Example API.
- **Oracle XE⁹:** Oracle Database 10g Express Edition (Oracle Database XE) is an entry-level, small-footprint database based on the Oracle Database 10g Release 2 code base that's free to develop, deploy, and distribute; fast to download; and simple to administer.
- **MySQL¹⁰:** A simple, yet powerful Open Source Software relational database management system that uses SQL.
- **MSSQL:** Microsoft SQL Server 2005¹¹ is comprehensive, integrated data management and analysis software that enables organizations to reliably manage mission-critical information and confidently run today's increasingly complex business applications. SQL Server 2005 allows companies to gain greater insight from their business information and achieve faster results for a competitive advantage.

9.1.4 Messaging

Provided as a part of Open ESB (see section 9.1.2)

9.1.5 Communication networks

Communication of information uses HL7 messages wrapped in SOAP envelopes transferred using HTTP over TCP-IP. In some cases, specific security measures must be taken, e.g. when communicating with restricted health networks.

9.1.6 Firewall Issues

Web Services uses SOAP over HTTP for communication. This implies that port 80 needs to allow for bidirectional communication.

⁸ <http://www.hibernate.org>

⁹ <http://www.oracle.com/technology/products/database/xe/index.html>

¹⁰ <http://www.mysql.com>

¹¹ <http://www.microsoft.com/sql/default.mspx>

10 Related work

10.1 Healthcare Service Specification Project (HSSP)

The HSSP project is a collaborative effort between Health Level Seven¹² and the Object Management Group¹³ to identify and document service specifications, functionality, and conformance supportive and relevant to healthcare IT stakeholders and resulting in real-world implementations. In addition, several other groups have joined the HSSP¹⁴ effort.

10.2 Open Healthcare Framework

The Eclipse Open Healthcare Framework (OHF)¹⁵ is a project within Eclipse formed for the purpose of expediting healthcare informatics technology. The project is composed of extensible frameworks and tools which emphasize the use of existing and emerging standards in order to encourage interoperable open source infrastructure, thereby lowering integration barriers. We currently provide tools and Frameworks for HL7, IHE, Terminology, Devices, and Public Healthcare Maintenance.

The OHF Project is currently in incubation. (October 2008)

¹² <http://www.hl7.org>

¹³ <http://www.omg.org>

¹⁴ <http://hssp.wikispaces.com>

¹⁵ <http://www.eclipse.org/ohf/>

References

1. Erl, T., *Service-Oriented Architecture Concepts, Technology, and Design*. The Prentice Hall Service-Oriented Computing Series ed. T. Erl. 2006, Crawfordsville, Indiana, USA: Prentice Hall.
2. OASIS Open, *Reference Model for Service Oriented Architecture 1.0*, C. Matthew MacKenzie, et al., Editors. 2006, OASIS Open.
3. University of Cyprus (UCY), *MPOWER D2.1 Relevant Standards and Sensors*. 2007, MPOWER Consortium, FP6 STREP 034707.
4. Austrian Research Center (ARC), *MPOWER D4.1 Interoperability Standards and Technology Overview*. 2007, MPOWER Consortium, FP6 STREP 034707.
5. University of Cyprus (UCY), *MPOWER D5.1 Security Standards and Technology Overview*. 2007, MPOWER Consortium, FP6 STREP 034707.
6. Gamma, E., *Design patterns: elements of reusable object-oriented software*. 1995.
7. Ali Arsanjani. *Service-oriented modeling and architecture: How to identify, specify, and realize services for your SOA*. 2004 [cited 2007 November 2]; Available from: <http://www-128.ibm.com/developerworks/webservices/library/ws-soa-design1/>.
8. Honey, A., A. Dutta, M. Kumar, and M. Christian, *SOA4HL7 Architecture Document*, A. Dutta, Editor. 2006, Health Level Seven. p. 76.
9. Stahl, T. and M. Völter, *Model-driven software development: technology, engineering, management*. 2006, Chichester: Wiley. XVI, 428 s.
10. Object Management Group (OMG), *MDA Guide Version 1.0.1*, J. Miller and J. Mukerji, Editors. 2003, Object Management Group. p. 1-62.
11. Miller, J. and J. Mukerji, *MDA Guide Version 1.0.1*, J. Miller and J. Mukerji, Editors. 2003, Object Management Group (OMG). p. 1-62.
12. Object Management Group (OMG), *UML 2.0 Superstructure FTF Rose model containing the UML 2 metamodel*. 2005, Object Management Group (OMG).
13. Mellor, S.J., *MDA Distilled: Principles of Model-Driven Architecture*. 2004.
14. Rosen, M., *MDA, SOA, and Technology Convergence*, in *The MDA Journal Straight from the Masters*, David S. Frankel and John Parodi, Editors. 2004, Meghan-Kiffer Press: Tampa, Florida, USA. p. 62-79.
15. Object Management Group (OMG), *Object Constraint Language (OCL), Version 2.0*. 2006, Object Management Group. p. 1-232.
16. Warmer, J.B., *"Object Constraint Language, The: Getting Your Models Ready for MDA, Second Edition"*. 2003.
17. CEN TC251, *EN 13940-1: Health Informatics - System of Concepts to Support Continuity of Care - Part 1: Basic Concepts*. 2006, European Committee for Standardization. p. 105.
18. World Wide Web Consortium (W3C), *Web Services Architecture*, D. Booth, et al., Editors. 2004, W3C.
19. World Wide Web Consortium (W3C), *Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language*, R. Chinnic, et al., Editors. 2007, W3C.
20. OASIS Open *Web Services Business Process Execution Language Version 2.0*, A. Alves, et al., Editors. 2007, OASIS.

Appendix A IBM Reference Architecture SOA

A.1 An architectural template for SOA

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

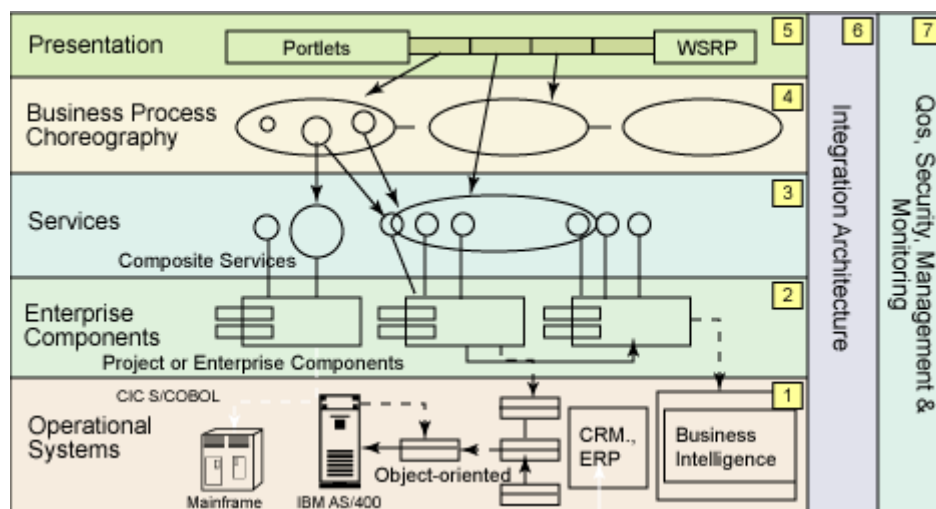


Figure 28 IBM SOA Reference Architecture

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.

A.2 IBM SOA Architectural template and MPOWER Platform

Following IBM SOA Architectural template we can define that **layers 2, 3, 6 and 7 are in the focus of the platform.** As MPOWER platform is build from scratches there are no existing legacy systems that need to be wrapped but however there are different sensors and actuators that need to be exposed as services independent of their vendor, type or communication standard and extern systems like HIS for which adapter will have to be made so that they can successfully communicate with MPOWER platform. We have called this layer “Physical layer”. The MPOWER architecture that follows the IBM SOA Reference architecture is presented on the next picture.

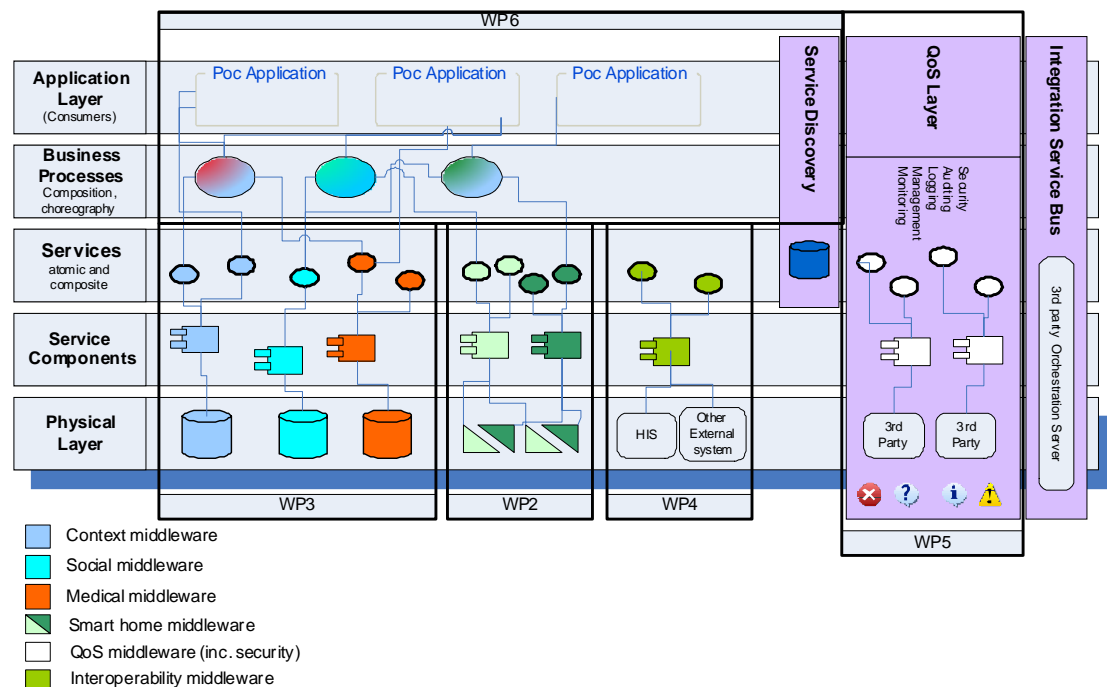


Figure 29 MPOWER Reference Architecture

A.3 Modelling styles using this reference architecture

Although IBM defines reference architecture for building SOA applications there are different ways how this applications can be modelled following the above mentioned architecture. The main modelling principles have to be defined for layers 2 and 3 that are closely connected. To solve this problem, Independent Software Vendor (ISV) partners are collaborating to create a specification for building systems that use service-oriented architecture. This specification is known as service-component architecture (SCA) specification and it briefly defines how service components are created, implemented and composed. The second and third layers that are presented on the following picture will be briefly described. This includes the description of the following terms: Service, Service component (or Enterprise component) and Composite (or Composite services).

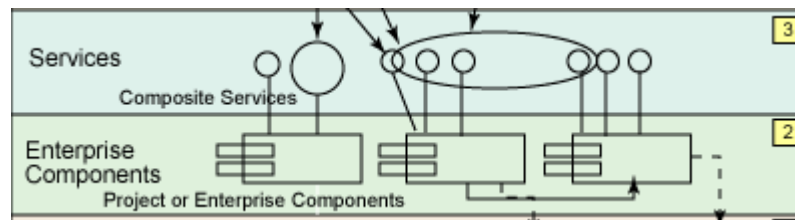


Figure 30 Cooperation between services and enterprise components

A.3.1 Component (or Service component or Enterprise component)

Following SCA specification, Component is the main unit of construction. A Component consists of a configured instance of an implementation, where an implementation is the piece of program code providing business functions. The business function is offered for use by other components as services. Implementations may depend on services provided by other components – these dependencies are called references. Implementations can have settable properties, which are data values which influence the operation of the business function. The component configures the implementation by providing values for the properties and by wiring the references to services provided by other components. The next picture presents a component and elements that are relevant for it.

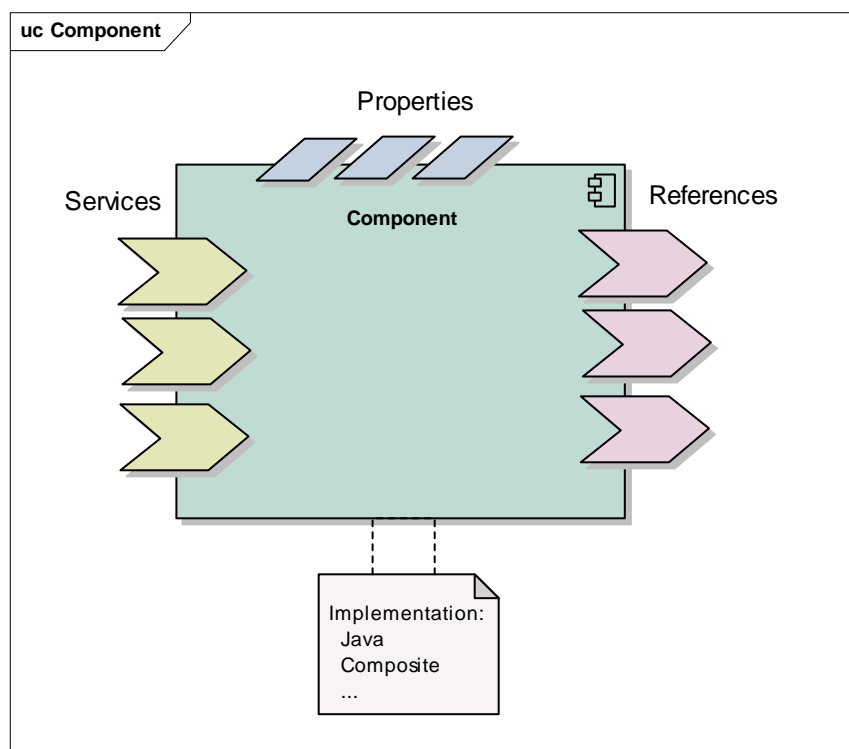


Figure 31 Explanation of component in SOA story

- Properties allow for the configuration of an implementation with externally set values.
- A Reference represents a requirement that the implementation has on a service provided by another component.
- A Service represents an addressable interface of the implementation.

A.3.2 Composite (or Composite services)

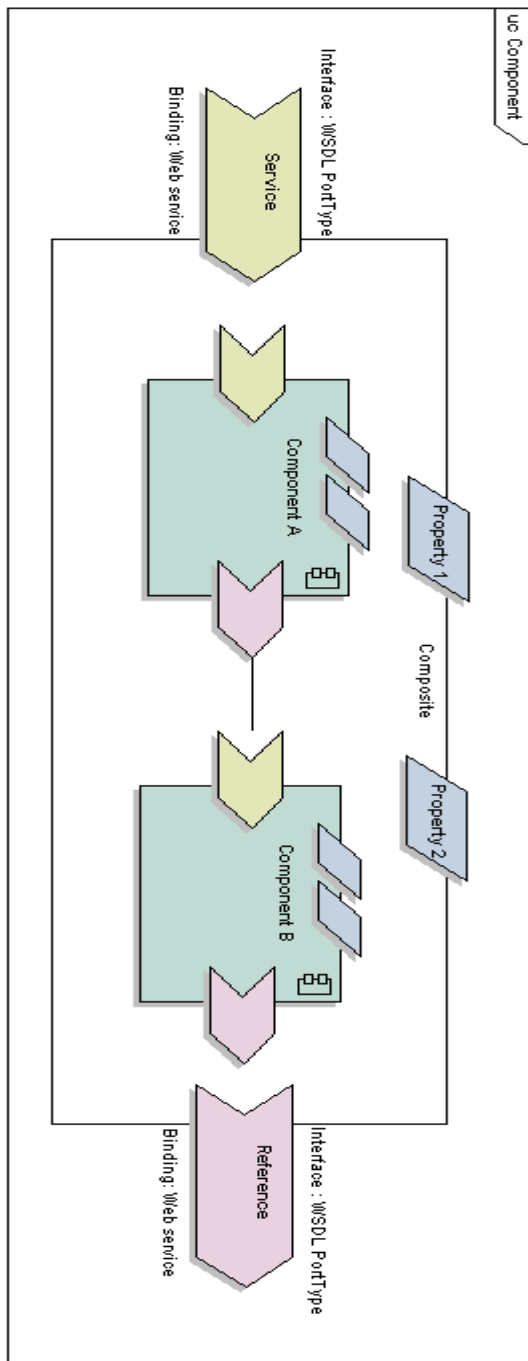


Figure 32 Way of building composite services

A composite contains a set of components, services, references and the wires that interconnect them, plus a set of properties which can be used to configure components. An composite is presented on the SOA reference architecture picture as an ellipse in the services layer. It composes different services with goal of providing new granular business functionality.

Appendix B UML Profiles for Homecare

The following text is from the article:

Walderhaug, S., E. Stav, and M. Mikalsen, *Experiences from model-driven development of homecare services: UML profiles and domain models*. in *2nd International Workshop on Model-Based Design of Trustworthy Health Information Systems (MOTHIS 2008)*. 2008. Toulouse: Springer

Experiences from model-driven development of homecare services: UML profiles and domain models

Ståle Walderhaug¹⁺², Erlend Stav¹ and Marius Mikalsen¹

{stale.walderhaug | erlend.stav | marius.mikalsen}@sintef.no,

¹SINTEF ICT, SP Andersens vei 15b, N-7465 Trondheim, Norway

²Department of Computer Science, University of Tromsø, N-9000 Tromsø, Norway

Abstract. Model-driven development approaches such as OMG's Model Driven Architecture (MDA) have been proposed as the new paradigm for software development. However, the adoption of MDA is still low, partly because of the general-purpose modelling language being used. Domain specific modelling languages are being developed for technological and industrial domains to improve the expressiveness and effect of model-driven development techniques. The healthcare domain could benefit from these methodologies to improve development speed and software quality. In order to incorporate domain knowledge in a MDA process, information about workflows, artefacts and actors can be formalized in a UML profile and applied by MDA tools for design and development. This paper presents the results from the work done on model-driven development of smart homecare services in the MPOWER project. Following an iterative approach, two UML profiles to support development of Service Oriented Architecture based homecare applications are proposed. The profiles are based on a comprehensive domain investigation and best practice methods for domain specific modelling language development. Using homecare specific UML profiles indicate an improvement in the process for model-driven development of homecare services, and more evaluation will be conducted in an experiment in 2009.

B.1 Introduction

Model-Driven Development (MDD) such as OMG's Model Driven Architecture (MDA) [10], has the potential to improve the quality of software systems. Quality attributes such as interoperability, reusability and appropriateness of software components and systems are main features of MDA. By using abstraction and advanced automation techniques, software artefacts are created from formal models that are represented using languages such as the Unified Modeling Language (UML) [35]. The core of the MDA process, and similar MDD processes, is to use formal models as the main development artefacts in the entire development process, from domain analysis to implementation, deployment and testing.

Domain specific modelling languages (DSML) have been proposed as a means to overcome many of the shortcomings with UML and MDA. The scientific knowledge about applying MDD techniques in design and development of healthcare information systems is scarce [36]. Creating DSMLs for the healthcare domain is a daunting task, and requires extensive investment of resources and time.

We set out to investigate how MDD with DSML support should be introduced and applied in a healthcare sub domain. In the MPOWER project [37], we have developed a framework for creating homecare software services using a model-driven approach. The framework defines a MDA toolchain which is a set of modelling, transformation and development tools that supports the complete MDA process as described in the MDA Guide [10]. A comprehensive model of actors and services in homecare along with the MDA toolchain for designing and implementing these domain specific web services has been developed and evaluated.

This paper presents research results from the project with focus on:

1. What is the domain knowledge in homecare that can be used as assets in the MDA process?
2. Which knowledge can be included in a UML Profile for homecare services and how can this knowledge be utilized by developers?

The MPOWER toolchain, providing model traceability, model transformation and code generation, has been evaluated in the development of two proof-of-concept applications and is currently being redesigned with improved UML Profile support for the domain. Based on the experience from developing the MPOWER framework and proof-of-concept applications a conceptual domain model and UML profile for service oriented computing in the homecare domain is proposed and discussed.

The rest of this paper is organized as follows. In the next section we describe the background for this work, including relations to and motivations for applying model driven development, domain specific modelling languages, and service oriented architecture. With this background we proceed to describe the method applied and main activities within the MPOWER project. The main results from each of the main activities are presented next, including conceptual domain models and our preliminary DSML approach based on two UML profiles. A discussion section follows this, before we conclude the paper.

B.2 Background and Related work

The work presented herein is a part of the EU-IST project MPOWER (contract no. 034707) and of an ongoing PhD thesis work by the main author. MPOWER is a user driven research and development project where the main goal is to create a framework for rapidly creating standards-based homecare services. The framework includes the definition of a toolchain which is being used in the development of two proof-of-concept applications targeting elderly and cognitive impaired people living at home.

MDD promises a potential to improve the quality of software systems and their development by using formal models as first class entities in the entire development process. When MDD is done properly, the result may include the following improvements in the development process:

- Domain experts, system architects and developers can discuss concepts and requirements more effectively because care workflow and concepts are modelled formally.
- Formal modelling of domain concepts allows for more precise understanding of the target system's requirements, environment and use scenarios, which will increase the probability for developing a software system that fulfils its intentions
- Analysis of system behaviour and performance can be conducted in the design phase, facilitating rapid feedback and agile development processes.
- Keeping models independent of implementation technology and deployment platforms as long as possible allows for reuse of software models, thus improving interoperability and reducing development cost.

Tuomainen et al argues that modelling helps the understanding of healthcare activities by being illustrative, identifying improvements, simulate organisational processes and individual activities in healthcare [38]. They compare three model centric approaches; MDA, Business Process Modelling with BPMN and BPEL and the HL7 development framework. They conclude that in order to realise their full potential these approaches require local and project specific adaptation. This paper explains such an adaptation for the homecare domain.

B.3 Methods

The main objective of the MPOWER project is to create a framework that facilitates rapid development of homecare services. To achieve this, it is fundamental to acquire knowledge about the homecare domain, and make this domain knowledge available to actors involved in the system development processes. Due to the complexity of the healthcare domain, it was considered imperative to iterate between domain modelling and system design. To facilitate this interaction, the MPOWER project defined three main activities:

1. Capture domain knowledge from experts on aging/dementia, healthcare workers in the domain, family carers and patients. The result will be a number of conceptual domain models defining the actors, actor relationships, core use cases and main system requirements / services of the homecare domain
2. Specify a MDA toolchain that support documentation of system requirements, modelling of design and development of services. Moreover, the toolchain must be evaluated in terms of usability and

- usefulness/performance by implementing two Proof-of-Concept Applications (PoCA). The results are a MDA toolchain with evaluation reports on developer acceptance and technical qualities
- Design a DSML that incorporates the domain knowledge from task 1 and MDA toolchain experience from task 2. The result will be one or more UML profiles that can be used with a revised MDA toolchain.

The main activities and the expected results are illustrated in Figure 33 and described in more detail in the following.

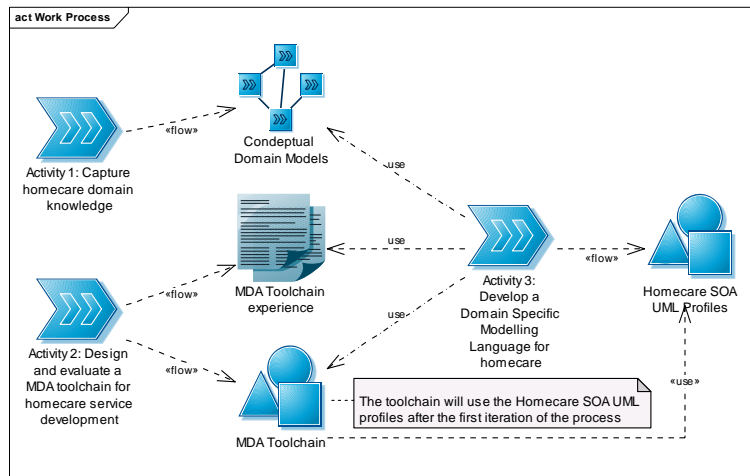


Figure 33 The three main project activities and their work products

B.3.1 Activity 1: Capture Domain Knowledge

The MPOWER project focuses on *smart homecare solutions for elderly and cognitive impaired people*. The domain models for the work being presented in this paper can be seen from two different viewpoints:

- The Homecare viewpoint: this viewpoint focuses on organizational aspects of homecare as well as the main stakeholders (people and systems) involved. This model is the result of a comprehensive process involving a total of 140 domain stakeholders such as domain experts, professional caregivers, family carers and patients [39, 40].
- The Homecare SOA viewpoint: this viewpoint focuses on the main system components and their relationships in terms of the principles of Service Oriented Architecture design. Important assets for this model are the design principles given by Erl [1], and SOA4HL7 methodology [41]. The structure and semantics of the domain model is supported by the SOA reference architecture from IBM along with the IBM UML profile for software services [21].

B.3.2 Activity 2: Designing a Toolchain for MDD in Homecare

To have a formal way of specifying the domain models, proper modelling tools are needed. In the beginning of the MPOWER project, a set of tools were specified as the MPOWER toolchain to be used by all involved personnel for conceptual modelling, requirements specification, analysis, system design, system development, deployment and testing. A single toolchain for all involved stakeholders ensures that information provided by domain experts is readily available for system designers using the same toolchain. The process used for selection of tools matches the recommendation given by Staron [42], page 240: “The process [of creating domain models] should be tool independent. The independence should be supported by using technologies that are open and unbounded, but at the same time supported by more than one tool.” The following tools were used:

- Microsoft Word 2003: Used for describing user scenarios. MS Word was available to all project participants involved in the process
- Enterprise Architect (EA) V6.5 from Sparx Systems: Used for UML modelling of use cases and services, model transformation and code generation of WSDL code.
- IBM’s UML 2.0 profile for Software Services: Used during modelling the services to structure models and stereotype core elements. Available from IBM [21].
- NetBeans V6.0: Used as Java IDE for generating service skeletons and implementing the services. Open source using open standards, <http://www.netbeans.org>

The described toolchain was used from the start of the project with only minor modifications such as EA upgrades and bug fixes. The two PoCAs were developed using the toolchain and the performance of the toolchain, were investigated from two perspectives: 1) Developer acceptance of the MPOWER toolchain: using the Technology Acceptance Model with two additional factors, as reported in [43], and 2) Technical review: Weekly scrum and quarterly technical meetings with workshop sessions on how to improve the toolchain.

B.3.3 Activity 3: Refine the MPOWER toolchain and develop a DSML

The UML standard allows for the creation of a DSML in two ways: 1) Creating a new language based on Meta Object Facility [32], or 2) extending UML through the use of UML Profiles. As discussed by Selic, the latter will often be the most practical and cost-effective solution [44], and is also the chosen method for this paper. By using UML profiles to create a DSML, the semantics and syntax of UML can be inherited, and powerful UML/MDA tools can be used with the profile for software development.

The UML Profile standard [35], outlines several reasons for creating a DSML from UML. The most pertinent reasons for the challenges addressed in this paper are:

- Terminology adapted to the healthcare domain
- Add information that can be used during transformation
- Add constraints that restrict the way you can use the metamodel

Despite the fact that DSMLs are becoming more and more popular in systems modelling, there is not much knowledge in the scientific community about best practices for creating DSMLs with UML [44, 45]. However, in a paper from 2007, Selic summarized the basic steps for creating a DSML in terms of UML in [44]:

1. Create a conceptual domain model: The model should include the essential concepts of the domain, the relationships between the concepts, the constraints that govern the use of the concepts. In addition, syntax and semantics of the notation should be provided. A selection of UML models from Activity 1 makes up the conceptual domain model.
2. Map domain model to a UML profile, refining the core UML specification with stereotypes, tagged values and constraints. In addition, a library of domain specific modelling elements (model library) can be defined for reuse.

The process of creating a domain specific UML profile is not straightforward, since the level of abstraction and the intended use of the profile play an important role for the definition of the profile elements. This challenge is tackled with experience from the design of the MPOWER toolchain and development of two MPOWER PoCAs for the homecare domain.

To identify and model the elements of a UML profile is an iterative process. To guide this process, the Staron's guidelines for defining good stereotypes using a classification schema [42], is used. This classification is based on the stereotype's role and expressiveness.

B.4 Results

The results presented in this section are based on the work carried out in the MPOWER project from October 2006 to June 2008.

B.4.1 Activity 1: Conceptual Domain models

The domain models were developed in several iterations from October 2006 to September 2007. Details about the process and findings from the user needs investigation from which the domain models are derived can be found in [39].

Figure 34 shows the main concepts from a homecare viewpoint. To keep the model at an abstract level and not overpopulate it with unnecessary details, most attributes on the classes are hidden. The main classes and relationships are:

- Subject of Care (SoC): This is the person receiving care through a homecare program. The SoC has a unique identifier that is managed by the assigned healthcare organization. A SoC must be associated with at least one healthcare professional.
- Homecare Program: a class comprising the services, devices and healthcare organizations involved in providing homecare service to a SoC.
- Carer: an individual that is a part of the family, a healthcare professional or a friend. All HealthcareProfessionals must be associated with a HealthcareOrganization.

- Healthcare Organization: an organisation that is directly involved in the provision of care to a SoC.
- Homecare Service: a service provided to the SoC through a Homecare program. Three core types of services have been identified: information service (e.g., calendar, educational material access, medication list), communication services (e.g. SMS, email), and assistive service (e.g. indoor location service, heating control, burglar alarm, oven control).

Concepts in the model are aligned with the concepts presented in Continuity of Care (CONTSYS) standard from CEN TC251 [26], and service categories from [46]. Most concepts are also available in the HL7 RIM, but CONTSYS is more specific than HL7. These resources were found useful in selecting an appropriate abstraction level and structure in the domain model. The complete models of actors and services are presented in [40].

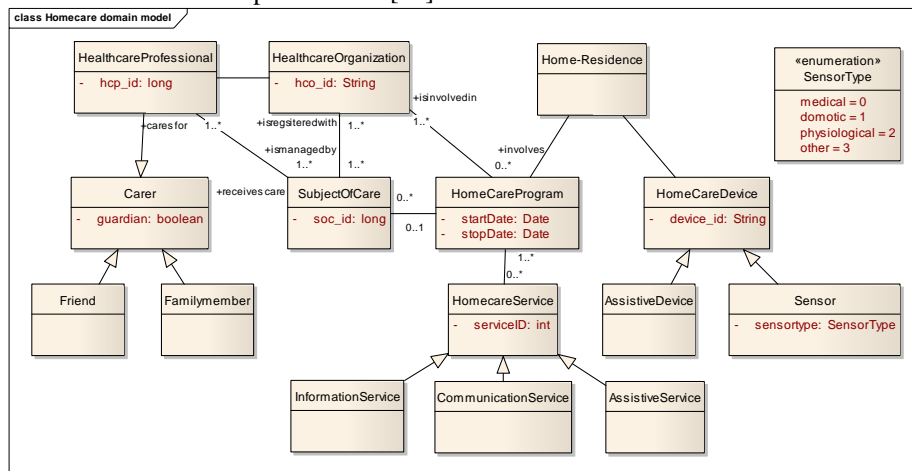


Figure 34 Diagram showing the main concepts in a smart homecare domain

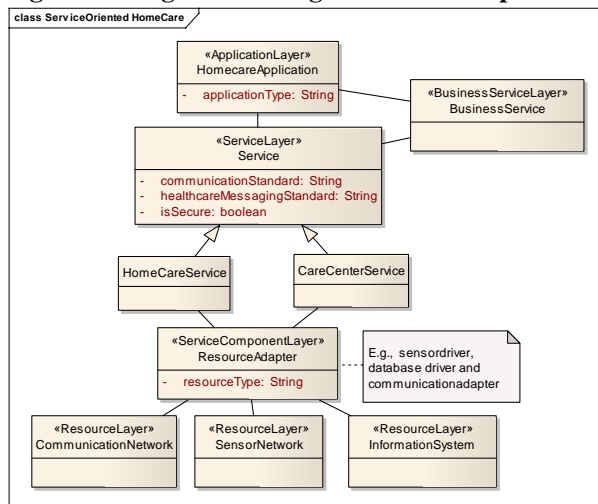


Figure 35 The Service-oriented view on a typical homecare environment

Figure 35 shows the main components, stereotyped with the five layers of the IBM SOA reference architecture.

B.4.2 Activity 2: The MPOWER Toolchain

The experiences from using the described toolchain for development of the services that form the PoCAs are grouped into developers’ subjective experience and technical experience. The first group entails perceived characteristics such as the factors described by the Technology Acceptance Model [47]. The results from a developer evaluation of the MPOWER toolchain is presented in [43], and concludes that perceived ease of use and perceived usefulness are factors that affect the developers’ adoption of MDA. It was also found that traceability between artefacts in the development process was useful. A major drawback with the evaluated toolchain was found to be the incomplete code generation features. A technical review of the MPOWER toolchain revealed that 1) WSDL model transformation incomplete: it was necessary to customize the transformation template for WSDL models, 2) WSDL code generation had errors: the built-in transformations in the EA tool generated some errors that had

to be changed manually, e.g. using “type” references instead of “element” references in message definitions, and 3) Performance problems using HL7: the import of HL7 message types into WSDL resulted in tool crashes because of memory allocation problems. Recursive import of HL7 xml schema (xsd) definitions were not handled by the WSImport tools in Netbeans.

From the experience with the MPOWER toolchain, a set of new features were proposed. The developers would like more support in generating the implementation of the services – repetitive code (e.g. for DB management, handling of security, return status), and support for object/relational persistence service, such as generation of Hibernate mappings for the information elements declared in the message definitions of the WSDLs. These features may impact the design of a DSML as they could require domain specific information to be incorporated into the models during the service design.

B.4.3 Activity 3: Refined Toolchain - Mapping of Domain Concepts to DSML - UML Profile

The process of defining a UML Profile for SOA in homecare use concepts from the conceptual domain models and experience from toolchain and PoCA development in an iterative approach. This section presents the preliminary results from Activity 3 after the first iteration (January-June 2008). Activity 3 is carried out by a core team of three researchers. However, some discussions are brought into the development team mailing list (16 developers). The profiles were updated in three main revisions: initial version for the start of the development, second version after first version of services and the third version after the first iteration of the PoCA development. The changes between version one and two were significant, whereas only tags and minor adjustments to relationships were done for version 3.

Two UML profiles are proposed, the Homecare UML Profile and the SOA Homecare UML Profile. The profiles can be used separately or together in a MDA development project, depending on how the profile elements are utilized in the development process.

B.4.4 Homecare UML profile

Figure 36 shows the Homecare UML profile. The profile elements are mainly derived from the homecare conceptual model (Figure 34). The tagged values on the stereotypes were identified based on experience from the MDA toolchain work. The mapping of concepts to the UML profile was also guided by the CONTSYS standard [26].

Table 3. Table describing the proposed stereotypes in and tagged values in the Homecare UML profile

Stereotype	Comment
SubjectOfCare	Subject of care (SoC) is defined in CONTSYS as “person seeking to receive, receiving, or having received health care” [26]. Used to decorate SoC modelling elements and to add information about the SoC that can be used during model transformation or code generation, e.g., to generate interface classes for EHR systems and individual plan systems. <i>SoC_type</i> : describes can be used to describe different types of SoC according to national or healthcare enterprise specific patient classifications.
Carer	A stereotype that should be used on all modelling elements representing an individual that provides care, professionals as well as non-professional caregivers. <i>roles</i> : can be used to set the default role, e.g. in terms of security, for the instances of classes marked with this stereotype.
Healthcare Professional (HcP)	Defined in CONTSYS as “person authorised by a nationally defined mechanism to be involved in the direct provision of certain health care activities” [26]. Should be used to mark all modelling elements of type Class/Actor that are representing individuals that fit this definition. The roles attribute is inherited from Carer.
Other Carer	Defined in CONTSYS as “person providing assistance for activities of daily living or social support”. This stereotype should be used to mark modelling elements of type Class/Actor that are representing individuals such as family members, friends and other carers employed by non-healthcare organizations such as home services

	and security services. The roles attribute is inherited from Carer
Healthcare Organization (HcO)	Defined in CONTSYS as “organisation involved in the direct provision of health care” [26]. This stereotype should be used to mark all modelling elements of type Class/Actor that represents organisations that fits the CONTSYS definition. <i>orgainsationType</i> : is used to describe the type of organisation according to speciality levels, private versus public or other national classifications.
Homecare Device	Generic homecare device stereotype to be used on modelling elements of type Class/Actor. The stereotype can be useful for design-time checking of interoperability and interconnectivity of devices in a homecare system. <i>deviceType</i> : describes the type of this device, e.g. whether it is medical or domotic. <i>interfaceType</i> : describes the kind of interface used to connect to this device. In the UML profile, the HomecareDeviceInterface enumeration is defined based on the experience in the MPOWER project. This enumeration includes the most used interface types, and can be refined to fit other technologies.
Healthcare Professional For SubjectOfCare	A stereotype that is used to mark an association between a HealthcareProfessional and a SubjectOfCare. The stereotype can be used to ensure that a SubjectOfCare is associated with at least one HealthcareProfessional.
EmployedAt	A stereotype that is used to mark an association between a HealthcareProfessional and a HealthcareOrganisation. The stereotype can be used to check that all HealthcareProfessional “types” are employed at a HealthcareOrganisation “type”.

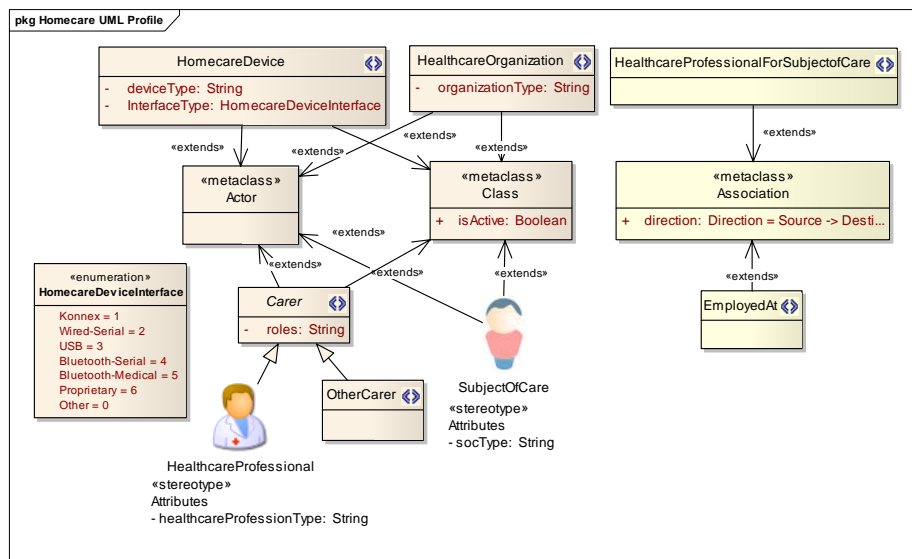


Figure 36 First version of Homecare UML Profile

All the stereotypes in the Homecare UML Profile falls into the category “Virtual Metamodel Extension, restrictive” defined by Staron in [42]. These are stereotypes that reuse the semantics of the metaclasses (e.g. Actor and Class). Often the must be used with other stereotypes, making the stereotyped model element more precise and may also add a new icon to the concrete syntax to familiarize the model presentation (e.g. icons on HealthcareProfessional and SubjectOfCare).

B.4.5 SOA Homecare UML Profile

The SOA Homecare UML profile enables developers to create precise models of SOA-based homecare systems.

Table 4. List of stereotypes and tagged values in the SOA Homecare UML Profile

Stereotype	Comment
Homecare Application	<p>A stereotype used to mark modelling elements of type Component and Class. The stereotype adds two properties to the modelling elements:</p> <p><i>securityLevel</i>: this describes the security level of the application. This can be used to check that a user (i.e., service, component, application) of the application must have at least the same access level in order to be allowed to use the service. For service composition, this information can be used to check that the application can access security-enabled services.</p> <p><i>applicationType</i>: this describe the type of application this is, e.g. in terms of deployment configurations.</p>
Homecare Service	<p>A service which is used in the homecare environment. The IBM Service Profile should be used in combination with this stereotype.</p> <p><i>securityLevel</i>: this describes the security level of the service. This allows for specification of the security requirements and rights for a modelling element that can be utilized during code generation. In the next version of the UML Profile, this stereotype will be updated with more security tags addressing service-service authorization and information encryption.</p>
Homecare Message	<p>To denote elements that are messages used in interactions of homecare services and applications.</p> <p><i>isPersistent</i>: indicates whether the message data is stored in a database or not. This can be used for creating Hibernate mapping code and database schema.</p> <p><i>messagingStandard</i>: the standard to which this message belongs. Can be used both for code generation, ensuring correct libraries are present, and for checking conformance with the standard.</p>
Assistive Service	<p>A type of homecare service that provides assistive functionality in the homecare system. Derived from Stefanov's classification for smart house services for elderly and cognitive disabled [46].</p>
Information Service	<p>An information service which will be used by stakeholders in a homecare setting. The service can be medically related, but can also be other services e.g. used for social interaction by the subject of care.</p> <p><i>serviceType</i>: defines the type of information service this service belongs to. An enumeration is proposed based on the experience in the MPOWER project. This enumeration can be refined in other projects using other kinds of information sources.</p>
UsesHomecare Message	<p>A stereotype that mark associations between a HomecareService and a HomecareMessage. The stereotype can be used to generate traceability information that can again be used when messaging standards are being updated or changed.</p>
UsesHomecare Service	<p>A stereotype that mark associations between a HomecareApplication and a HomecareService. The stereotype can be used to generate traceability information that again can be used when a homecare service is being updated or changed.</p>

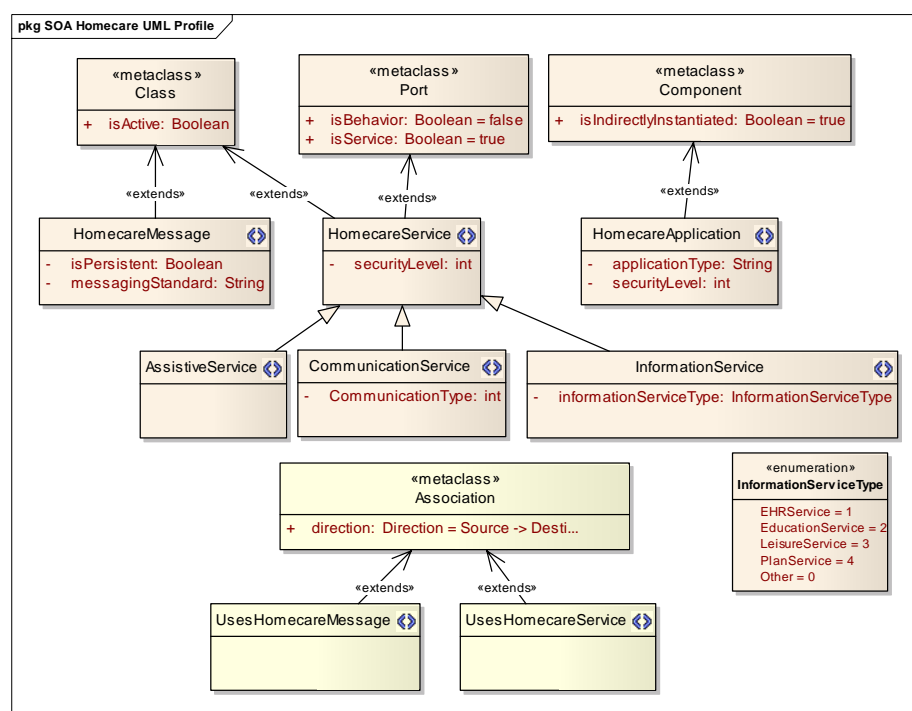


Figure 37 The Homecare SOA UML Profile diagram

All stereotypes in the Homecare SOA UML profiles falls into the category “Code generation, restrictive” defined by Staron [42]. These are stereotypes that extend the base metaclass (e.g., Class and Port) with some properties to increase the precision of the semantics and restrict the usage.

B.5 Discussion

The work presented in this article is a part of an initiative to develop a model-driven software development framework for healthcare, focusing on homecare services in the first phase. It is considered imperative to incorporate domain knowledge into the framework and make this knowledge readily available for architects and developers in all development phases. This paper presents the results from creating a domain specific modelling language for homecare using UML Profiles.

Capturing the conceptual domain knowledge for homecare, or any other healthcare sub-domain, is a daunting task. Many stakeholders are involved, as well as a plethora of information systems, involving many different coding standards and vocabularies. These factors, in addition to legislative factors and organisational aspects, make modelling of reusable healthcare domain models difficult, but not impossible. To succeed in creating a useful model-driven software development process, it is important to choose the right level of abstraction and in divide the healthcare application areas into well defined sub-domains.

The homecare domain model shown in Figure 34 shows the most important actors and relationships between them. The model would fit for modelling most homecare solutions, is aligned with the CEN TC251 CONTSYS standard [26], and includes the main classifications from Stefanov’s acknowledged paper on smart house technologies for elderly [46]. The model is the result of a comprehensive domain analysis process where 140 domain stakeholders from four European countries were involved in improving the validity of the model [39]. If new concepts are developed for the domain, these can be added as an extension to the existing homecare model, without compromising the original model and the related UML profiles.

The Homecare SOA model is based on the domain investigation from the MPOWER project, in addition to the IBM SOA reference model and IBM UML profile for software services [21]. The Homecare SOA model provides information about deployment of services and possible configuration and information sources. The model is on an abstract level, and could in certain cases be refined with details about security platforms and network connectivity details. Such domain knowledge could be useful in planning the distribution of services and integration with existing resources, but will also make the model less suitable for reuse across different healthcare enterprises and nations.

To make the domain knowledge readily available as assets in the development process, UML profiles were chosen, inline with the recommendations by Selic [44]. UML profiles builds upon the

syntax and semantics of UML, and most UML tools support profiles. This is an imperative advantage, enabling developers to use their favourite UML tool for design and development. The process of selecting domain concepts to include as stereotypes, tagged values or constraints in a UML model, requires knowledge about model-driven development, but also experience from modelling systems in the domain in question. Experience from the development of a MDA toolchain (Activity 2) provided information about which target software artefacts that should be generated from the models and which models and diagrams that should be applied for achieving this. This information was of utmost importance when choosing the metaclass extensions for the elements in the UML profiles.

Model-driven development seeks to represent all development artefacts as modelling elements. UML profiles are used to customize the modelling language to include familiar concepts that enables more effective and precise system design and implementation. Stereotypes, the main UML profile mechanism, can be used for several purposes, as discussed by Staron [42]. The result from the mapping of domain concepts to a DSML (Activity 3) showed that all stereotypes in the Homecare UML profile are classified as Virtual Metamodel Extensions. This implies that this profile is mainly used to increase the expressiveness of the modelling language when designing systems for homecare. A “virtual model extension restrictive” stereotype, adds a domain specific icon such as a picture of a nurse to the modelling element, together with a well known domain specific label such as HealthcareProfessional.

The Homecare SOA UML Profile includes elements from the “Code generation, restrictive” category. This means that the main use of these stereotypes is to improve code generation by providing domain information so that code generation scripts can create high-quality code. In this paper code generation was restricted to WSDL and Hibernate code, but other software artefacts can be generated from the domain information in a UML profile. Test cases, error-checking code, security policy verification, and result validation may be generated if the required information is available. Examples are:

- During design of a homecare service, mark the service class with HomecareService, add classes for the messages that will be used as input and output, mark these with HomecareMessage and set the tagged value MessagingStandard to the appropriate value. Create an association between the service and the message by applying the UsesHomecareMessage stereotype on the association. During code generation, the generator could look up messaging details (syntax, namespaces, etc) in an external resource and insert this code into e.g., the WSDL types definitions.
- When modelling the information models for a homecare system, mark the patient class with SubjectOfCare stereotype. To ensure that a patient is associated with a healthcare professional, a model validator based on OCL [23], can iterate through the patient class’s relationships and look for a HealthcareProfessionalforSubjectOfCare stereotyped association.

The two proposed UML profiles can be used on the same models to provide different “views”. In addition, the IBM Software Service UML profile should be used to complement the service design models for SOA Homecare systems.

B.6 Concluding remarks

Model-driven development approaches can be improved by extending the modelling language with domain specific concepts. UML Profiles can be used as a mechanism for toolchains based on OMG’s MDA and UML standards.

The UML Profiles must provide information that can improve the design and/or code generation processes. The two profiles proposed in this paper are based on solid work on capturing homecare domain knowledge and experience from developing homecare SOA systems using MDA. Though the profiles are still undergoing updates and improvements, they can improve modelling process performance and results.

Another finding from the process is that it is useful to have a development activity in parallel with specification of the conceptual domain and profile. The experience from this development activity gave valuable input to the mapping of concepts to DSML processes. This finding extends the proposed approaches to DSML development in [44, 45, 48].

The MPOWER Toolchain will be evaluated by university students in 2008. In 2009, an experiment measuring the subjective improvements (perceived characteristics) and objective improvements (e.g., reduction of errors, time spent for development) will be conducted with 20 developers from the healthcare domain.

References

1. Erl, T., *Service-Oriented Architecture Concepts, Technology, and Design*. The Prentice Hall Service-Oriented Computing Series ed. T. Erl. 2006, Crawfordsville, Indiana, USA: Prentice Hall.
2. OASIS Open, *Reference Model for Service Oriented Architecture 1.0*, C. Matthew MacKenzie, et al., Editors. 2006, OASIS Open.
3. University of Cyprus (UCY), *MPOWER D2.1 Relevant Standards and Sensors*. 2007, MPOWER Consortium, FP6 STREP 034707.
4. Austrian Research Center (ARC), *MPOWER D4.1 Interoperability Standards and Technology Overview*. 2007, MPOWER Consortium, FP6 STREP 034707.
5. University of Cyprus (UCY), *MPOWER D5.1 Security Standards and Technology Overview*. 2007, MPOWER Consortium, FP6 STREP 034707.
6. Gamma, E., *Design patterns: elements of reusable object-oriented software*. 1995.
7. Ali Arsanjani. *Service-oriented modeling and architecture: How to identify, specify, and realize services for your SOA*. 2004 [cited 2007 November 2]; Available from: <http://www-128.ibm.com/developerworks/webservices/library/ws-soa-design1/>.
8. Honey, A., et al., *SOA4HL7 Architecture Document*, A. Dutta, Editor. 2006, Health Level Seven. p. 76.
9. Stahl, T. and M. Völter, *Model-driven software development: technology, engineering, management*. 2006, Chichester: Wiley. XVI, 428 s.
10. Object Management Group (OMG), *MDA Guide Version 1.0.1*, J. Miller and J. Mukerji, Editors. 2003, Object Management Group. p. 1-62.
11. Miller, J. and J. Mukerji, *MDA Guide Version 1.0.1*, J. Miller and J. Mukerji, Editors. 2003, Object Management Group (OMG). p. 1-62.
12. Object Management Group (OMG), *UML 2.0 Superstructure FTF Rose model containing the UML 2 metamodel*. 2005, Object Management Group (OMG).
13. Mellor, S.J., *MDA Distilled: Principles of Model-Driven Architecture*. 2004.
14. Rosen, M., *MDA, SOA, and Technology Convergence*, in *The MDA Journal Straight from the Masters*, David S. Frankel and John Parodi, Editors. 2004, Meghan-Kiffer Press: Tampa, Florida, USA. p. 62-79.
15. Object Management Group (OMG), *Object Constraint Language (OCL), Version 2.0*. 2006, Object Management Group. p. 1-232.
16. Warmer, J.B., *"Object Constraint Language, The: Getting Your Models Ready for MDA, Second Edition"*. 2003.
17. CEN TC251, *EN 13940-1: Health Informatics - System of Concepts to Support Continuity of Care - Part 1: Basic Concepts*. 2006, European Committee for Standardization. p. 105.
18. World Wide Web Consortium (W3C), *Web Services Architecture*, D. Booth, et al., Editors. 2004, W3C.
19. World Wide Web Consortium (W3C), *Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language*, R. Chinnic, et al., Editors. 2007, W3C.
20. OASIS Open *Web Services Business Process Execution Language Version 2.0*, A. Alves, et al., Editors. 2007, OASIS.
21. Object Management Group (OMG), *UML 2.1.2 Superstructure and Infrastructure*. 2007, Object Management Group (OMG).
22. Mohagheghi, P. and V. Dehlen, *Where Is the Proof?-A Review of Experiences from Applying MDE in Industry*. Model-Driven Architecture-Foundations and Applications: 4th European Conference, Ecmda-Fa 2008, Berlin, Germany, June 9-13, 2008, Proceedings, 2008.
23. MPOWER Consortium. *Middleware platform for eMPOWERing cognitive disabled and elderly*. 2006 [cited 2007 January 17]; Available from: <http://www.sintef.no/mpower>.

24. Tuomainen, M., et al. *Model-centric approaches for the development of health information systems*. in *Medinfo*. 2007. Brisbane, Australia.
25. Holthe, T., et al., *Analysing user needs prior to technology development for supporting older people and people with cognitive impairments at home. Experiences from the MPOWER-project*. International Journal of Medical Informatics, 2008. **submitted July 2008**.
26. Walderhaug, S., E. Stav, and M. Mikalsen, *Reusing models of actors and services in smart homecare to improve sustainability*. Stud Health Technol Inform, 2008. **136**: p. 107-12.
27. Honey, A. and B. Lund, *Service Oriented Architecture and HL7 v3: Methodology*. 2006, HL7 Service Oriented Architecture Special Interest Group (SOA SIG). p. 79.
28. Johnston, S. *UML 2.0 Profile for Software Services*. 2005 [cited 2007 June 15]; Available from: http://www.ibm.com/developerworks/rational/library/05/419_soa/.
29. Staron, M., *Improving modeling with UML by stereotype-based language customization*, in *School of Engineering*. 2005, Blekinge Institute of Technology: Blekinge. p. 270.
30. Walderhaug, S., et al. *Factors affecting developers' use of MDS in the Healthcare Domain: Evaluation from the MPOWER Project*. in *From code-centric to model-centric development, Workshop at European Conference on Model-Driven Architecture*. 2008. Berlin, Germany: European Software Institute.
31. Object Management Group (OMG), *MOF 2.0 / XMI Mapping Specification, v2.1*. 2005.
32. Selic, B., *A Systematic Approach to Domain-Specific Language Design Using UML*. 10th IEEE ISORC, 2007. **7**.
33. Lagarde, F., et al., *Improving uml profile design practices by leveraging conceptual domain models*. Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering, 2007: p. 445-448.
34. Stefanov, D.H., Z. Bien, and W.-C. Bang, *The smart house for older persons and persons with physical disabilities: structure, technology arrangements, and perspectives*. IEEE transactions on neural systems and rehabilitation engineering, 2004. **12**(2): p. 228-50.
35. Davis, F.D., R.P. Bagozzi, and P.R. Warshaw, *User Acceptance of Computer Technology: A Comparison of Two Theoretical Models*. Management Science, 1989. **35**(8): p. 982-1003.
36. Fuentes-Fernández, L. and A. Vallecillo-Moreno, *An Introduction to UML Profiles*. UML and Model Engineering, 2004. **5**(1).

Appendix C MDS and Interoperability

The following text is from the article:

S. Walderhaug, M. Mikalsen, G. Hartvigsen, E. Stav, J. Agedal, "Improving systems interoperability with model-driven software development for healthcare," *Medinfo*, vol. 12, no. Pt 1, 2007, pp. 122-126.

Improving Systems Interoperability with Model-Driven Software Development for HealthCare

Ståle Walderhaug^{a+b}, Marius Mikalsen^a, Gunnar Hartvigsen^b, Erlend Stav^a, Jan Agedal^a

^a SINTEF ICT, Trondheim, Norway

^b Medical Informatics and Telemedicine Group, Dept for Computer Science, University of Tromsø, Tromsø, Norway

C.1 Abstract:

An aging population and an increase in chronically ill patients demand teamwork treatment models. To support these with information systems, interoperability is a prerequisite. Model-driven software development (MDS) with special healthcare extensions can enable reuse of components and improve conformance to international standards. In this paper, a MDS HealthCare Framework is proposed and demonstrated for homecare services. Using the framework, information systems will improve their conformance to international standards and the interoperability with other systems.

C.2 Introduction

In healthcare, the disease burden is changing from acute to chronic care, 35000000 people died from chronic diseases in 2005, and 60% of all deaths are due to chronic diseases [1]. New ways of providing care are being evaluated, based on teamwork treatment – demanding support from interoperable information systems. Interoperability in healthcare has been identified as an important area of research and development by many organizations, including the European Union (EU)¹⁶, the Object Management Group (OMG)¹⁷ and other national organizations [2]. The ability to exchange information and share services across departmental, organizational and national borders can reduce the administrative overhead and costs [3], and as a result improve the effectiveness of healthcare provided. Consequently, more patients can be treated faster with the same amount of (care) resources. A sustainable healthcare infrastructure depends upon interoperable health information services [4, 5].

The treatment and management of homecare consumers, typically elderly, chronically ill and cognitive disabled, require a coordinated effort from healthcare and social welfare services. To effectively support these care services with information systems, interoperability of core information such as patient careplan calendar and medication-list is a prerequisite.

¹⁶ EU Life sciences, genomics and biotechnology for health website: <http://cordis.europa.eu/lifescihealth/home.html>

¹⁷ Object Management Group (OMG) website: <http://www.omg.org>

To improve interoperability between systems, the leading standardization bodies in healthcare information, HL7, CEN TC251 and OpenEHR, have specified standards that address systems architecture and information exchange. Although these standards have been available to the Health Information Systems (HIS) vendors for some time, they have not fully adopted them into their products. Thus, the different HIS are not interoperable, requiring the development of software adapters to be able to exchange information about the patients. There is an urgent need for a standardized interface and method to realize this information exchange.

The standardization bodies provide limited tool support to the developers of health information systems. To incorporate standard healthcare concepts in the systems' design, an operational software engineering artefact that provides both semantic and syntactic interoperability functionality [6, 7] should be available for the system architects and developers [8-10].

In 2002, the Object Management Group (OMG) introduced the Model-Driven Architecture (MDA) [11], an approach focusing on using models (e.g., UML models [12]) as first-class entities in the development of software systems. In practice, this means that the models are used directly in the implementation of an information system, either as system blueprints or as input to code generation engines that produce executable code. MDA is the most known model-driven software development (MDSD) approach, and the overall idea is to separate business functions (in Platform Independent Models - PIM) from its technological implementations (in Platform Specific Models – PSM), enabling code generation and reuse of components. The overall benefit is improved interoperability and reduced development time and cost.

Using a MDSD approach in the development of healthcare information system services could facilitate the use of standards through specification of reusable standards-based PIMs. Advanced UML mechanisms such as Profiles and Patterns could be used to further extend the expressiveness of the modeling language and force the use of standardized healthcare concepts. As a result, the developed systems will increase the level of interoperability, and at the same time development and maintenance costs will decrease.

With an aging population and a rapidly increasing number of chronically ill patients [1], the need for teamwork treatment is crucial. Healthcare Information Systems (HIS) can no longer be seen as standalone systems, but need to interoperate in a health network [10]. This leads to the problem statement: *How can health information systems development be improved to ensure that systems involved in a homecare teamwork treatment infrastructure can share information in an effective and sustainable manner?*

This paper proposes a model-driven software development framework with standards-based healthcare extensions as a tool to achieve interoperability between HIS. The healthcare focus is on homecare services although the healthcare standards discussed have general applicability. The paper concludes that MDSD with the appropriate healthcare information extensions can improve software's conformance to standards and thus also the ability for caregivers to share information in teamwork treatment.

Following next is an overview of the challenges that are associated with developing such a MDSD Healthcare framework, both from a software engineering and healthcare viewpoint. Then the framework is presented along with an example from the homecare domain, before the paper concludes with a discussion of the validity of our results and directions for future work.

C.3 Immature MDSD tools and need for evaluations

In a keynote talk at the 2006 ECMDA-FA conference in Bilbao (Spain), Bran Selic (IBM) advertised for rigorous scientific studies that investigate how MDSD can improve the development process¹⁸. Recently, the ModelWare project¹⁹ conducted five different scientific MDSD evaluations. A summary of the evaluations is presented in [13] and concludes that by applying MDSD, a productivity gain of 20% can be expected and the quality of the software produced would increase.

Despite these and other reports, there is a considerable skepticism in the software engineering community about the performance and usability of MDSD. The skepticism is based on three main points: 1) the UML is too generic and is conceptually too far from implementation languages making it

¹⁸ ECMDA website: <http://www.ecmda-fa.org/>

¹⁹ MODELWare (FP6-IP 511731) project website: <http://www.modelware-ist.org>

difficult to generate efficient and fully executable code [14, 15], 2) The maturity of MDSD tools: transformation tools are not complete enough to provide return of the investment put into developing reusable UML models. E.g., the Query/View/Transformation (QVT) standard [16] by OMG does not have good tool support and 3) standards are used in different versions, some of which are not interoperable.

C.4 Many Systems, Many Standards

The use of information standards to improve interoperability between information systems in the healthcare domain is not straight-forward. In a single healthcare organization, there is a plethora of information systems, each based on one or more information standards. In the context of systems development, sharing of information and services between these systems need to address the following issues: 1) Many systems (such as patient administrative systems) are dated back to the late eighties, long before the specification of today's information standards, 2) Department specific systems developed to serve one specific purpose do often not use international standards nor follow best-practice in systems architecture, 3) The information systems themselves and the information standards used are continuously being upgraded [7].

C.5 A MDSD Framework for HealthCare

The work presented in this paper build upon three assertions presented in the following.

Assertion 1: Model-Driven Software Development with healthcare information standards support will improve interoperability between health information systems (compared to the traditional way of developing systems)

UML allows for extensions through the use of UML Profiles. A profile defines stereotypes, tagged values and constraints that can be assigned to modeling elements in the design process. The main purpose of a profile is to extend UML's expressiveness for a certain domain, e.g. healthcare. By providing healthcare specific UML profiles and patterns as a part of a MDSD framework for healthcare, concepts defined in international healthcare information standards can be automatically built into the information systems. A healthcare profile can be used by transformation templates and code generators to explicitly implement attributes, relationships, operations and objects that provide interoperability services.

Assertion 2: Healthcare Information Standards are appropriate as reusable model-driven development artefacts.

Standards from HL7, CEN TC251 and OpenEHR make use of UML class diagrams to specify concepts and relationships. However, parts of the semantics are described textually as constraints-comments to the formal UML models. To be able to correctly incorporate these standards into model-driven development artefacts such as UML Profiles, the complete semantics of the standards must be possible to represent formally. The correctness and reusability of the models created with the UML profile will depend on the mapping between the standard and the UML profile artefacts.

Assertion 3: Healthcare information services in the homecare domain can be reused across organizations.

The usefulness of a MDSD Healthcare framework for the development of interoperable homecare services will depend on the ability to define functional and coherent information services in the domain. The services need to be reusable beyond departmental and organizational borders, preferably also national borders as some healthcare institutions have rehabilitation and treatment centers abroad, often collaborating with the local healthcare services.

C.6 Results

Using a model-driven approach such as the MDSD Healthcare Framework enables rapid development of interoperable healthcare information systems. The framework includes a set of UML profiles, models and experience reports from the homecare domain, but with generic healthcare service applicability.

C.6.1 Example of MDSD Healthcare Framework in Homecare

A trivial example is provided to demonstrate how a UML Profile for healthcare can be used in the development process to achieve interoperability between information systems.

The example service is a CarePlan service where a HomeCare Center System and a General Practitioner (GP) EHR HomeCare extension can access and update the homecare patient’s careplan. Both systems will need to provide a defined interface for information exchange based on the same standard. A small subset of the “CarePlan” concept in the Continuity of Care (CONTSYS) [17] standard is used for demonstration (Figure 38). A “CarePlan” is applied by one or more HealthCare Professional and addresses one or more health issues that the Subject of Care has (relation not shown).

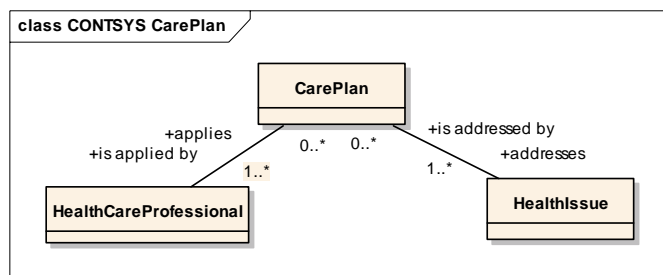


Figure 38: A subset of the CONTSYS CarePlan concept

The goal is to develop Java based (sub-) systems that allow exchange of careplan information for the homecare patient according to the CONTSYS standard.

C.6.2 A Simple UML Profile for HomeCare

Based on the CONTSYS standard, the following UML extensions are specified: 1) UML Class Stereotype: SubjectOfCare: The person receiving treatment, 2) UML Class Stereotype: CarePlan: The treatment plan for one or more health issues (problem), 3)UML Class Stereotype: HealthCareProfessional: A caregiver entitled to provide care, 4) UML Association Stereotype: HealthCareProfessional_isResponsible: The healthcare professional (source element) is responsible for the target element and 5) UML Association Stereotype: SubjectOfCare_Owns: Subject of Care (source) has owner right of the target element.

Two tagged values are defined: 1) Boolean: isShared: when used with a CarePlan, stating whether the careplan is shared or not and 2) Boolean: isOrganDonor: used with a SubjectOfCare to state if the person is organ donor or not.

C.6.3 The Healthcare Information Systems

The two systems are being developed independently by different vendors using the same CONTSYS-based UML profile. The Care Center system platform independent model (PIM) shown in Figure 39 shows that the HomeCarePlan (stereotyped CarePlan) is related to the HomeCarePatient (owned by), the Doctor (under responsibility of) and the Visiting Nurse. All classes are stereotyped according to CONTSYS. As a result, the HomeCarePatient has a tagged value for “isOrganDonor” and the HomeCarePlan has an “isShared” tag.

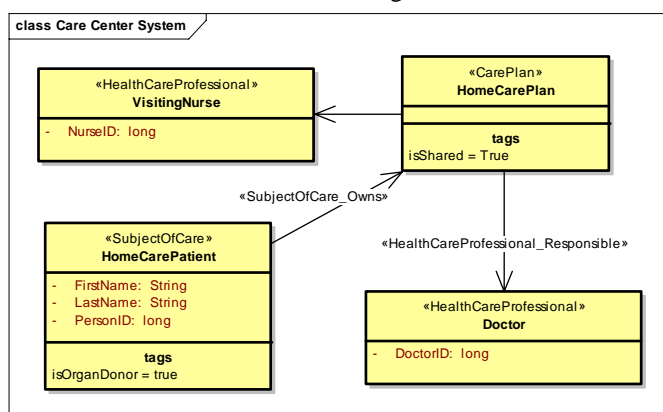


Figure 39: The Care Center System PIM

The PIM for the GP EHR Homecare extension system (Figure 40) shows that the TreatmentPlan (“CarePlan”) elements are related to one or more patient problems (“HealthIssue”) according to a problem-oriented EHR [18]. This can be used to filter out treatment activities that are not related to the coordinated care of a homecare patient.

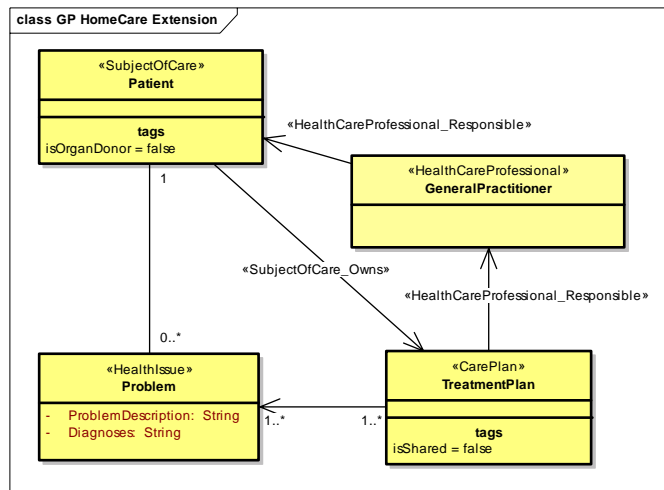


Figure 40: The GP EHR Homecare Extension PIM

The two PIMs can be transformed to Java Platform Specific Models (PSM) using a CONTSYS-based transformation script for Java. This script utilizes the stereotypes and tagged values in the transformation process to add attributes and operations to ensure that the required interoperability mechanisms are implemented. In this trivial example, only set and get operations for the tagged values and careplan elements are created. The Java Model for the Care Center system (Figure 41) and the GP EHR Homecare extension (Figure 42) show that during the transformation process, three operations have been created on the CarePlan-stereotyped classes. These operations, stereotyped with “CarePlan”, enables exchange of CarePlan elements and retrieval of all HealthCare Professionals that are related to the CarePlan.

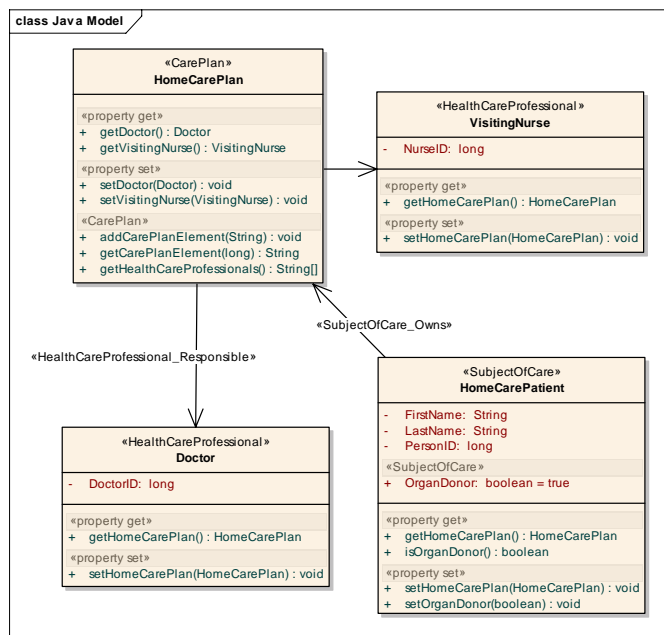


Figure 41: Java PSM for the Care Center System

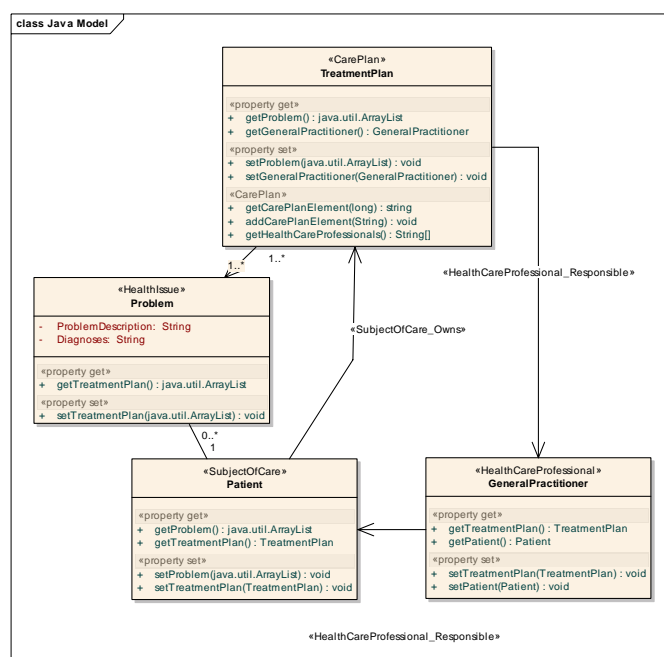


Figure 42: Java PSM for the GP EHR HomeCare Extension

From these Java PSMs, code can be generated using a standard code generation tool based on e.g. QVT [16].

To summarize: using the CONTSYS UML Profile in the design and development of the careplan service in the Care Center and GP EHR systems ensured that the services are conform to the standard and thus can exchange information correctly.

C.7 Discussion

The MDSO Healthcare framework proposed in this paper addresses the need to make information systems in the healthcare domain interoperable and sustainable. To achieve this, the framework provides tools and reusable components that incorporate international information standards into the information system design.

The effect this will have for the future healthcare information systems relies on the three assertions described in the first part: 1) the quality of artefacts produced from the framework, 2) the ability to map information standards to useful UML profiles and 3) the identification of reusable services.

The quality of the software produced by the framework will depend on the tool support and the developer. The main improvement compared to traditional software development lies in the built-in healthcare tool support, where use of healthcare UML profiles, reuse of existing platform independent models and use of code generation will reduce roundtrip time and improve the quality of the code.

The example showed a simple careplan service where a concept from CONTSYS was applied. More complex services will need more concepts, maybe from more than one standard. The MDSO framework will provide UML Profile support for the most used healthcare standards and patterns for the most recurring concepts. A modular design, in line with Beale's *archetype* concept (7), provides scalability and maintainability of the models as the standards are updated or extended. The *Archetypes* being specified in both CEN TC251 EN13606 [19] and OpenEHR, can be used by the MDSO framework as reusable models and patterns. An archetype is a model of a healthcare concept, and is represented formally using UML.

The specification of reusable services in the healthcare domain is in accordance with Service-Oriented Architecture (SOA) [20]. Many healthcare organizations are adapting SOA as the core enterprise architecture, using a message-oriented middleware with HL7 to exchange information between systems. The process of transitioning to a SOA architecture is expensive, but a fully interoperable healthcare infrastructure would reduce coordination expenses dramatically [3]. Homecare services are likely to be a part of this enterprise service architecture connected through a health network [5]. SOA-based homecare system services can enable independent development and deployment of new patient monitoring and surveillance services in the health network. A SOA based infrastructure will allow sustainable development of healthcare services.

A critical aspect when introducing new development tools and techniques is to evaluate its effect. Proper scientific methods must be applied to achieve rigor. A complete medical informatics solution should not only evaluate the artefacts isolated, but also study their effect in a real environment [21]. The MDSD HealthCare Framework will be subject for two scientific experiments with real users in the M-Power project²⁰.

C.8 Future work

The framework proposed in this paper is a part of the work being done within the M-Power and Linkcare projects²¹. These projects will identify and develop reusable homecare services for the provision and coordination of homecare services. Using the first version of the HealthCare MDSD framework, some of these services will be evaluated in 2007 and 2008.

C.9 Conclusion

With an aging population and dramatic increase in chronic diseases [1], systems interoperability in the healthcare domain is of utmost importance in order to maintain the service level of today and support teamwork treatment. One way to improve interoperability is to ensure the healthcare information systems' conformance to international standards.

The Healthcare MDSD framework will incorporate standards into the development process of information systems, and as a result improve interoperability. The MDSD framework will be evaluated in two experiments in 2007 and 2008 as a part of the LinkCare and M-Power projects. These projects have a strong focus on treatment and management services for chronically ill, elderly and cognitive disabled. This will ensure the framework's relevance for the domain.

C.10References

- [1] World Health Organization. Preventing CHRONIC DISEASES -a vital investment: World Health Organization; 2005.
- [2] Norwegian Ministry of Social Affairs, Norwegian Ministry of Health. Te@mwork 2007 - Electronic Interaction in the Health and Social Sector: Directorate for Health and Social Affairs; 2004.
- [3] Walker J, Pan E, Johnston D, Adler-Milstein J, Bates W. D, Middleton B. The Value of Health Care Information Exchange and Interoperability. Health Tracking. 2005 January;5(10).
- [4] Brailer D. Interoperability: the key to the future health care system. Health Affairs vol:Suppl Web Excl-19 2005.
- [5] Beyer M, Kuhn KA, Meiler C, Jablonski S, Lenz R. Towards a flexible, process-oriented IT architecture for an integrated healthcare network. Proc. of the 2004 ACM symposium on Applied Computing. Nicosia, Cyprus: ACM Press; 2004.
- [6] Park J. Information systems interoperability: What lies beneath? ACM transactions on information systems. 2004;22(4):595.
- [7] Beale T. Archetypes: Constraint-based Domain Models for Future-Proof Information Systems. OOPSLA 2002 Workshop on behavioural semantics; 2002; Portland, Oregon, USA; 2002.
- [8] Kuhn KA, Lenz R, Elstner T, Siegele H, Moll R. Experiences with a generator tool for building clinical application modules. Methods of information in medicine. 2003;42(1):37-44.
- [9] Lenz R, Beyer M, Kuhn KA. Semantic integration in healthcare networks. International journal of medical informatics. 2007 Feb-Mar;76(2-3):201-7.
- [10] Lenz R, Kuhn KA. Towards a continuous evolution and adaptation of information systems in healthcare. International journal of medical informatics. 2004;73(1):75-89.
- [11] Miller J, Mukerji J. MDA Guide Version 1.0.1: Object Management Group (OMG); 2003 2003-06-13. Report No.: omg/2003-06-01.

²⁰ MPOWER homepage: <http://www.mpower-project.eu>

²¹ LinkCare homepage: <http://www.linkcare-bcn.org>

- [12]Object Management Group (OMG). UML 2.0 Superstructure FTF Rose model containing the UML 2 metamodel: Object Management Group (OMG); 2005.
- [13]Hartman A. Industrial ROI, Assessment, and Feedback - Master Document: IBM Haifa Research Lab; 2006 September 15, 2006. Report No.: D5.3-1.
- [14]Fowler M. UML Modes. 2003 [cited 2006 November 14]; Martin Fowler's homepage]. Available from: <http://www.martinfowler.com/bliki/UmlMode.html>
- [15]Mellor SJ. Executable UML: A Foundation for Model-Driven Architecture; 2002.
- [16]Object Management Group (OMG). MOF QVT Final Adopted Specification: Object Management Group (OMG); 2005.
- [17]CEN TC251. EN 13940-1: Health Informatics - System of Concepts to Support Continuity of Care - Part 1: Basic Concepts: European Committee for Standardization; 2006 September 26.
- [18]van Bommel JH, Musen MA. Handbook of medical informatics. 1 ed: Springer Verlag; 2002.
- [19]CEN TC251. EN 13606-1-4: Health informatics - Electronic healthcare record communication Part 1-4: European Committee for Standardization; 2000 May.
- [20]OASIS Open. Reference Model for Service Oriented Architecture 1.0: OASIS Open; 2006 August 2. Report No.: soa-rm-cs.
- [21]Friedman CP. Where's the science in medical informatics? Journal of the American Medical Informatics Association. 1995;2(1):65-7.

Address for correspondence

Ståle Walderhaug, University of Tromsø, Department of Computer Science, Medical Informatics and Telemedicine group, 9037 TROMSØ, NORWAY. Telephone: +47 90766069, Fax: +47 77644580, email: stale.walderhaug@sintef.no

Appendix D Reusable Actors and Services

This appendix is from the paper presented at Medical Informatics Europe 2008:

S. Walderhaug, E. Stav, and M. Mikalsen. “Improving systems interoperability with model-driven software development for healthcare,” *Medinfo*, vol. 12, no. Pt 1, 2007, pp. 122-126.

Reusing Models of Actors and Services in Smart Homecare to Improve Sustainability

– Ståle Walderhaug^{a+b}, Erlend Stav^a and Marius Mikalsen^a
^a *SINTEF ICT, Trondheim, Norway*
^b *Department of Computer Science, University of Tromsø, Norway*

Abstract: Industrial countries are faced with a growing elderly population. Homecare systems with assistive smart home care technology enable elderly to live independently at home. Development of such smart home care systems is complex and expensive and there is no common reference model that can facilitate service reuse. This paper proposes reusable *actor* and *service models* based on a model-driven development process where end user organizations and domain healthcare experts from four European countries have been involved. The models, specified using UML can be reused actively as assets in the system design and development process and can reduce development costs, and improve interoperability and sustainability of systems. The models are being evaluated in the European IST project MPOWER.

Keywords: Homecare, Smart house, Service Oriented Architecture, System architecture, UML, Model-driven software development, standardisation, HL7

D.1 Introduction

Home care is a concept where new technological solutions allow the elderly to live independently at home. The consumers, typically elderly, chronically ill and cognitive disabled are empowered by state-of-the-art information and technology in their homes to achieve the overall goal of aging in place [49-51].

The technological advances are being deployed into the homecare domain, and many projects are working on *smart house* technology [46, 52] for homecare. This concept needs, in addition to technical devices, to be supported by a team of actors including family members, healthcare personnel and non-healthcare services organizations. A smart homecare system is both complex and expensive to build. Efficient development of complex systems should seek to *reuse* components and services through abstraction to “*manage complexity and guarantee continuity*” [53]. Krueger states that “*in order to reuse artefacts, software developers must either be familiar with the abstractions a priori or must take time to study and understand the abstractions.*” [54]. The importance of understanding the domain concepts is emphasized by Beyer et al in [55] where they state that “*to reduce the effort for system evolution it is highly desirable to incorporate generic components, that can be reused in different contexts*”.

Lenz, Beyer and Kuhn elaborate on this in [56] and argues for a separation of domain concepts and system implementation: “*in order to cope with domain evolution, modelling of domain concepts should be separated from IT system implementation. IT systems should be implemented by IT experts and medical knowledge should be modelled and maintained by domain experts.*” A reusable reference model for *smart homecare* can form the basis for more efficient development of smart homecare systems and reuse of its services [19, 55, 57, 58]. This article presents the work done in the MPOWER project [59] where a model of the core business process actors and services in smart homecare systems has been specified. Two research questions are addressed:

- Which actors (persons and systems) are involved in the teamwork treatment of smart homecare service consumers?
- Which information services are needed to support the (treatment) processes?

The focus of the work is on actors and information service support for elderly and cognitive disabled. The project result is a formal representation of actors and services that enable service reuse and increase the understanding of actor-service dependencies.

The remainder of the paper is organized as follows: First the methods and materials for specifying the actor and service models are described. Next, the core of these models is presented before the implications they may have on the development of sustainable healthcare systems are discussed.

D.2 Methods and Materials

Domain experts and user groups for elderly and cognitive disabled in four European countries were involved in the requirements process; Austria, Poland, the Netherlands and Norway [60]. The process produced the “User Scenario Specification” describing the problems experienced by the target groups and the planned assistive smart homecare services. Figure 43 shows the overall iterative approach for the work.

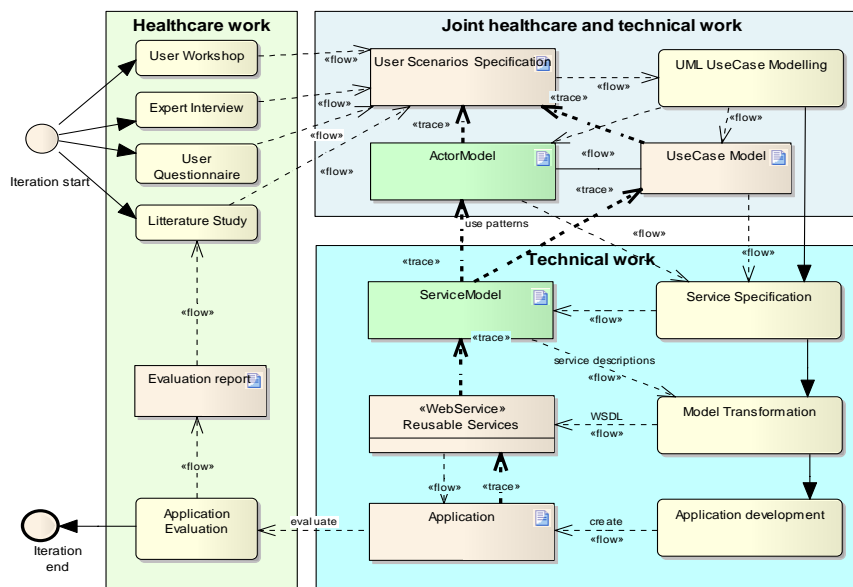


Figure 43: The iterative model-driven development process used to identify actors and services. Artefacts are shown as rectangles whereas the activities are denoted as rectangles with rounded corners.

This user requirements document was used as the basis for UML [11] UseCase Modelling activity that produced the UML Actor and UseCase Models. The modelling work was a joint effort by healthcare personnel and system architects, as recommended by Lenz et al in [56]. The Service Specification activity uses the Actor and UseCase models as input to create the Service Model which is used in the Model Transformation [10, 14, 61] activity for creating Reusable Services (Web Services). Finally, the Reusable Services are used by application developers to implement applications that will be evaluated by healthcare personnel. The evaluation report (from Application Evaluation activity) is used as input for the succeeding iteration. The bold arrows in Figure 43 highlight traceability links between artefacts [62].

The Use Case Model, Actor Model and Service Model were developed using Unified Modelling Language (UML) with the IBM Profiles for Software Services [21] according to Service Oriented Architecture (SOA) [1] concepts. The services in the Service Model were identified according to best practice SOA in general [1], and for healthcare especially [7, 41].

D.3 Results

In this paper, only a selection of the most important elements from these models is presented in detail. Full specifications are accessible on the MPOWER Website [59]. The use case and service modelling activities resulted in three UML models: the Actor Model, the Service Model, and the Use Case model that relates the two first.

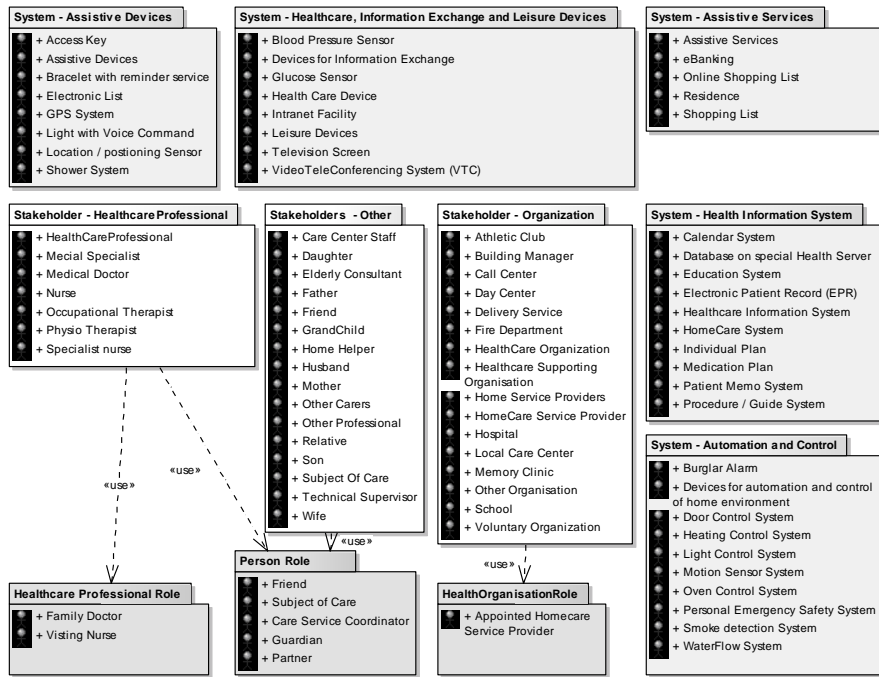


Figure 44: The ActorModel showing the elements of the system, stakeholder and role parts.

The Actor Model (Figure 44) has three main parts: System (light grey), Stakeholders (white) and Roles (dark grey). The stakeholders can have different roles as shown in Figure 45. The roles that the stakeholders can take are modelled as a dependency link, e.g., only a Nurse or a Specialist Nurse can have the role as a Visiting Nurse. All Healthcare Professionals can be a patient themselves (role Subject of Care).

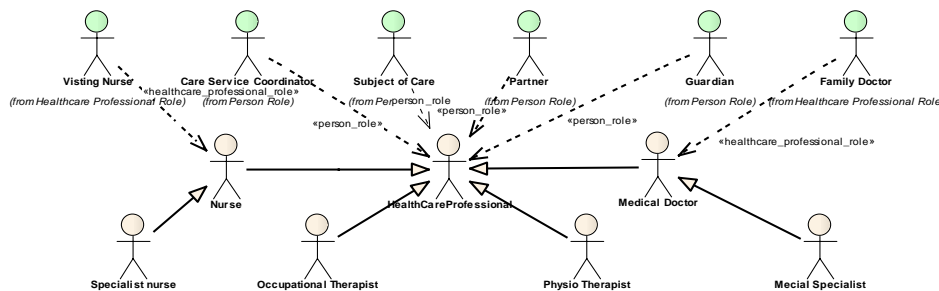


Figure 45: The relationships between healthcare professional actors and roles as defined in the ActorModel

The UseCase model defines activities that the actors (and roles) participate in. These activities are the link to the services in the ServiceModel. Figure 46 shows a use case diagram for calendar management activities involving systems, stakeholders and roles.

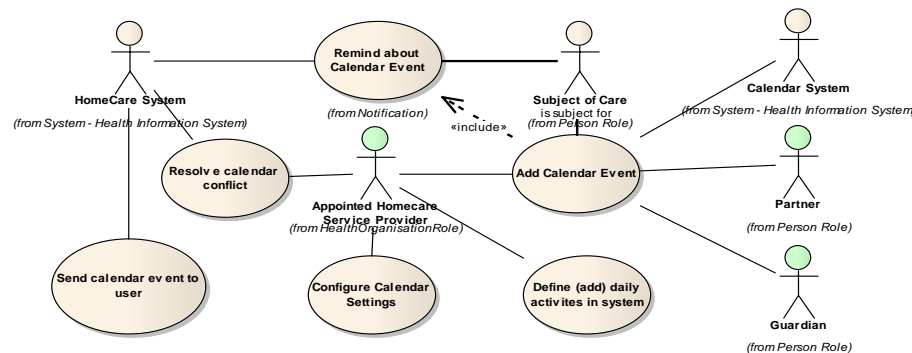


Figure 46: Actors, Roles and Use cases in the usecase diagram for Calendar Management services

From the scenarios and use cases five categories of services were specified (Figure 47) using the service identification principles described in [41].

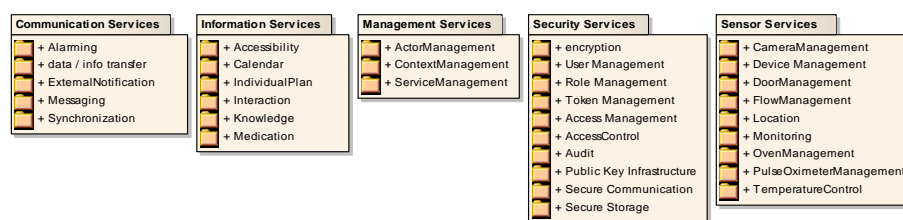


Figure 47: The service categories in the ServiceModel. The elements of each category are reusable services

D.4 Discussion

The Actor Model specified from the User Services Specification includes formal actor and role specifications in UML that allows for reuse across functional domains such as medication management and home automation services. A common and formally specified Actor Model means that 1) Actor-System (service) interaction can be precisely specified, 2) access control for services can be derived from use cases with actor interaction, 3) Actor-System dependencies can be traced through trace links, and 4) Systems and services can be compared to each other.

The elements of the Actor Model were agreed by all four countries participating in the specification process, and it is conform to the CEN TC251 CONTSYS [26] standard and compatible with those actors described in [63] and [46]. An important finding when modelling the Actor Model was the feasibility of using *roles* in use case modelling. In many cases, it is not the *actor* itself that interacts with the system, rather an actor taking a certain role. Using this concept, the constraints will be put on which *actor* that can take a *role* instead of modelling use cases for different actors in different contexts. Figure 46 shows an example where the *Appointed Homecare Service Provider*, *Partner* and *Guardian* roles are used to interact with the add calendar event activity.

The services identified for the Service Model are being implemented and used in two proof-of-concept applications in the MPOWER project [59]. The applications communicate with the services through HL7 messages, and all underlying complexity is handled by the services. The results from these developments will evaluate the services' reusability in the domain. Preliminary development results show that:

- Applications can be developed more rapidly by reusing high-quality services [55].
- Functionality can be reused across applications and organisations and nations, e.g., sending SMS, PKI, calendar management, medication list management.
- Clearly defined actors will improve the validity of the system services being developed and improve sustainability[53, 64]
- The gap between business processes and supporting information systems can be shortened by applying the Service Oriented Architecture concepts [1, 7, 19, 41]
- Aligning service and actor descriptions with national and international standards will promote standardisation and facilitate reuse of services and components across organisations and nations, thus improving interoperability[65]

Reuse of software and design is not trivial. Krüger states that “*for a software reuse technique to be effective, it must reduce the cognitive distance between the initial concept of a system and its final executable implementation.*” The Actor and Service models presented herein are the results of a formal process to reduce this distance.

The results presented herein are generalisations, and may not be directly applicable to all domains without prior local adaptations. The actor and service models can serve as reference models from which nation and organization specific models can be developed in accordance with the prevailing ways of organizing care and legislations. The specification of reference models must be supported by standards developing organisations such as CEN TC251 and HL7 [66]. The proposed models being implemented and evaluated in the MPOWER Project [59], and will be presented for the HSSP project [67] and national standardisation bodies in Europe.

D.5 References

1. Erl, T., *Service-Oriented Architecture Concepts, Technology, and Design*. The Prentice Hall Service-Oriented Computing Series ed. T. Erl. 2006, Crawfordsville, Indiana, USA: Prentice Hall.

2. OASIS Open, *Reference Model for Service Oriented Architecture 1.0*, C. Matthew MacKenzie, et al., Editors. 2006, OASIS Open.
3. University of Cyprus (UCY), *MPOWER D2.1 Relevant Standards and Sensors*. 2007, MPOWER Consortium, FP6 STREP 034707.
4. Austrian Research Center (ARC), *MPOWER D4.1 Interoperability Standards and Technology Overview*. 2007, MPOWER Consortium, FP6 STREP 034707.
5. University of Cyprus (UCY), *MPOWER D5.1 Security Standards and Technology Overview*. 2007, MPOWER Consortium, FP6 STREP 034707.
6. Ali Arsanjani. *Service-oriented modeling and architecture: How to identify, specify, and realize services for your SOA*. 2004 [cited 2007 November 2]; Available from: <http://www-128.ibm.com/developerworks/webservices/library/ws-soa-design1/>.
7. Honey, A., et al., *SOA4HL7 Architecture Document*, A. Dutta, Editor. 2006, Health Level Seven. p. 76.
8. Stahl, T. and M. Völter, *Model-driven software development: technology, engineering, management*. 2006, Chichester: Wiley. XVI, 428 s.
9. Object Management Group (OMG), *MDA Guide Version 1.0.1*, J. Miller and J. Mukerji, Editors. 2003, Object Management Group. p. 1-62.
10. Miller, J. and J. Mukerji, *MDA Guide Version 1.0.1*, J. Miller and J. Mukerji, Editors. 2003, Object Management Group (OMG). p. 1-62.
11. Object Management Group (OMG), *UML 2.0 Superstructure FTF Rose model containing the UML 2 metamodel*. 2005, Object Management Group (OMG).
12. Hartman, A., *Industrial ROI, Assessment, and Feedback - Master Document*. 2006, IBM Haifa Research Lab. p. 24.
13. Guttman, M. and J. Parodi, *REAL-LIFE MDA: Solving business problems with model driven architecture*. 2006: Morgan Kaufmann Publishers Inc, San Francisco, CA, USA. 200.
14. Mellor, S.J., *MDA Distilled: Principles of Model-Driven Architecture*. 2004.
15. Blobel, B.B. and P.P. Pharow, *A model-driven approach for the german health telematics architectural framework and the related security infrastructure*. Studies in health technology and informatics, 2005. **116**: p. 391-6.
16. Rubin, K.S., T. Beale, and B. Blobel, *Modeling for Health Care, in Person-Centered Health Records*. 2005, Springer New York: New York. p. 125-146.
17. Jones, V., A. Rensink, and E. Brinksma. *Modelling mobile health systems: an application of augmented MDA for the extended healthcare enterprise*. 2005.
18. Raistrick, C., *Applying MDA and UML in the Development of a Healthcare System*, in *UML Modeling Languages and Applications*. 2005, Springer Berlin / Heidelberg. p. 203-218.
19. Kawamoto, K. and D.F. Lobach, *Proposal for Fulfilling Strategic Objectives of the U.S. Roadmap for National Action on Decision Support through a Service-oriented Architecture Leveraging HL7 Services*. Journal of the American Medical Informatics Association, 2007. **14**(2): p. 146-155.
20. Omar, W.M., *E-health support services based on service-oriented architecture*. IT professional, 2006. **8**(2): p. 35.
21. Johnston, S. *UML 2.0 Profile for Software Services*. 2005 [cited 2007 June 15]; Available from: http://www.ibm.com/developerworks/rational/library/05/419_soa/.
22. Rosen, M., *MDA, SOA, and Technology Convergence*, in *The MDA Journal Straight from the Masters*, David S. Frankel and John Parodi, Editors. 2004, Meghan-Kiffer Press: Tampa, Florida, USA. p. 62-79.
23. Object Management Group (OMG), *Object Constraint Language (OCL), Version 2.0*. 2006, Object Management Group. p. 1-232.
24. Warmer, J.B., *"Object Constraint Language, The: Getting Your Models Ready for MDA, Second Edition"*. 2003.
25. Gamma, E., *Design patterns: elements of reusable object-oriented software*. 1995.

26. CEN TC251, *EN 13940-1: Health Informatics - System of Concepts to Support Continuity of Care - Part 1: Basic Concepts*. 2006, European Committee for Standardization. p. 105.
27. Mikalsen, M., et al. *Linkcare - interoperability accross levels and profession*. in *MEDINFO 2007*. Brisbane, Australia: IOS Press.
28. HL7. *HL7 Reference Information Model 2.16*. 2007 [cited 2007 June 22]; Available from: http://www.hl7.org/Library/data-model/RIM/modelpage_mem.htm.
29. World Wide Web Consortium (W3C), *Web Services Architecture*, D. Booth, et al., Editors. 2004, W3C.
30. SUN Microsystems. *Java Platform, Enterprise Edition 5 (Java EE 5)*. 2007 [cited 2007 June 22]; Available from: <http://java.sun.com/javaee/technologies/javaee5.jsp>.
31. World Wide Web Consortium (W3C), *Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language*, R. Chinnic, et al., Editors. 2007, W3C.
32. Object Management Group (OMG), *MOF 2.0 / XMI Mapping Specification, v2.1*. 2005.
33. HL7. *Health Level Seven (HL7)*. [cited 2007 June 22]; Available from: <http://www.hl7.org>.
34. OASIS Open Web Services Business Process Execution Language Version 2.0, A. Alves, et al., Editors. 2007, OASIS.
35. Object Management Group (OMG), *UML 2.1.2 Superstructure and Infrastructure*. 2007, Object Management Group (OMG).
36. Mohagheghi, P. and V. Dehlen, *Where Is the Proof?-A Review of Experiences from Applying MDE in Industry*. Model-Driven Architecture-Foundations and Applications: 4th European Conference, Ecmada-Fa 2008, Berlin, Germany, June 9-13, 2008, Proceedings, 2008.
37. MPOWER Consortium. *Middleware platform for eMPOWERing cognitive disabled and elderly*. 2006 [cited 2007 January 17]; Available from: <http://www.sintef.no/mpower>.
38. Tuomainen, M., et al. *Model-centric approaches for the development of health information systems*. in *Medinfo. 2007*. Brisbane, Australia.
39. Holthe, T., et al., *Analysing user needs prior to technology development for supporting older people and people with cognitive impairments at home. Experiences from the MPOWER-project*. International Journal of Medical Informatics, 2008. **submitted July 2008**.
40. Walderhaug, S., E. Stav, and M. Mikalsen, *Reusing models of actors and services in smart homecare to improve sustainability*. Stud Health Technol Inform, 2008. **136**: p. 107-12.
41. Honey, A. and B. Lund, *Service Oriented Architecture and HL7 v3: Methodology*. 2006, HL7 Service Oriented Architecture Special Interest Group (SOA SIG). p. 79.
42. Staron, M., *Improving modeling with UML by stereotype-based language customization*, in *School of Engineering*. 2005, Blekinge Institute of Technology: Blekinge. p. 270.
43. Walderhaug, S., et al. *Factors affecting developers' use of MDS in the Healthcare Domain: Evaluation from the MPOWER Project*. in *From code-centric to model-centric development, Workshop at European Conference on Model-Driven Architecture*. 2008. Berlin, Germany: European Software Institute.
44. Selic, B., *A Systematic Approach to Domain-Specific Language Design Using UML*. 10th IEEE ISORC, 2007. **7**.
45. Lagarde, F., et al., *Improving uml profile design practices by leveraging conceptual domain models*. Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering, 2007: p. 445-448.
46. Stefanov, D.H., Z. Bien, and W.-C. Bang, *The smart house for older persons and persons with physical disabilities: structure, technology arrangements, and perspectives*. IEEE transactions on neural systems and rehabilitation engineering, 2004. **12**(2): p. 228-50.

47. Davis, F.D., R.P. Bagozzi, and P.R. Warshaw, *User Acceptance of Computer Technology: A Comparison of Two Theoretical Models*. Management Science, 1989. **35**(8): p. 982-1003.
48. Fuentes-Fernández, L. and A. Vallecillo-Moreno, *An Introduction to UML Profiles*. UML and Model Engineering, 2004. **5**(1).
49. Magnusson, L.H., E.; Borg, M, *A literature review study of information and communication technology as a support for frail older people living at home and their family carers*. Technology and disability, 2004. **16**(4).
50. Wacanta, J.e.a., *Number of dementia sufferers in Europe between the years 2000 and 2050*. European Psychiatry, 2003. **18**: p. 306-313.
51. Demiris, G., et al., *Older adults attitudes towards and perceptions of smart home technologies: a pilot study*. Medical Informatics and the Internet in Medicine, 2004. **29**: p. 87-94.
52. Helal, S.C., *The Gator Tech Smart House: A Programmable Pervasive Space*. Computer, 2005. **38**(3): p. 50.
53. Shaw, M., *Abstraction techniques in modern programming languages*. IEEE Software, 1984. **1**(4): p. 10-26.
54. Krueger, C.W., *Software reuse*. ACM Computing Surveys (CSUR), 1992. **24**(2): p. 131-183.
55. Beyer, M., et al., *Towards a flexible, process-oriented IT architecture for an integrated healthcare network*, in *Proceedings of the 2004 ACM symposium on Applied computing*. 2004, ACM Press: Nicosia, Cyprus.
56. Lenz, R., M. Beyer, and K.A. Kuhn, *Semantic integration in healthcare networks*. International journal of medical informatics, 2007. **76**(2-3): p. 201-207.
57. Lenz, R., M. Beyer, and K.A. Kuhn, *Semantic integration in healthcare networks*. International journal of medical informatics, 2006.
58. Lenz, R. and K.A. Kuhn, *Towards a continuous evolution and adaptation of information systems in healthcare*. International journal of medical informatics, 2004. **73**(1): p. 75-89.
59. The MPOWER Consoritum. *MPOWER Website*. 2007 [cited 2007 June 15]; Available from: <http://www.mpower-project.eu>.
60. Prazak, B., et al. *User Requirements as Crucial Determinants for the Development of New Technological Solutions for Elderly Care - Exemplified in an European Project*. in *AAATE*. 2007. San Sebastian, Spain: IOSPress.
61. Frankel, D., *The MDA Journal: Model Driven Architecture Straight from the Masters*. 1st ed. 2004, Tampa, Florida, USA: Meghan-Kiffer Press. 219.
62. Walderhaug, S., et al., *Traceability in Model-Driven Software Development*, in *Designing Software-Intensive Systems: Methods and Principles*, P. Tiako, Editor. 2008, IDEA Group: Langston.
63. Toivanen, M., et al., *Gathering, Structuring and Describing Information Needs in Home Care: A Method for Requirements Exploration in a "Gray Area"*. MEDINFO 2004: Building High Performance Health Care Organizations, 2004: p. 7-11.
64. Coiera, E. and E.J. Shovenga, *Building a Sustainable Health System*, in *IMIA Yearbook of Medical Informatics 2007: Biomedical Informatics for Sustainable Health Systems*, A. Geissbuhler, R. Haux, and C. Kulikowski, Editors. 2007, Schattauer Publishers. p. 250.
65. Mykkeänen, J., *Specification of Resuable Integration Solutions in Health Information Systems*. Department of Computer Science. Vol. Doctoral Dissertation. 2007, Kuopio, Finland: Kuopio University Publications, Finland 133.
66. Blobel, B., *Authorisation and access control for electronic health record systems*. International Journal of Medical Informatics, 2004. **73**(3): p. 251-257.
67. HSSP Project, *The HSSP Roadmap: HSSP, Version 1.0*. 2007, Joint HL7-OMG Healthcare Services Specification Project. p. 13.

