

Die approbierte Originalversion dieser Dissertation ist an der Hauptbibliothek der Technischen Universität Wien aufgestellt (<http://www.ub.tuwien.ac.at>).

The approved original version of this thesis is available at the main library of the Vienna University of Technology (<http://www.ub.tuwien.ac.at/englweb/>).



TECHNISCHE  
UNIVERSITÄT  
WIEN

VIENNA  
UNIVERSITY OF  
TECHNOLOGY

Diese Dissertation haben begutachtet: .....

DISSERTATION

# ICT Support for Applying Evidence Based Tele Wound Management in a Distributed Semantic Web Environment

ausgeführt zum Zweck der Erlangung des akademischen Grades  
eines Doktors der technischen Wissenschaften unter der Leitung  
von

Univ.-Prof. Dr. Michael Binder  
722  
Universitätsklinik für Dermatologie  
Medizinische Universität Wien

eingereicht an der Technischen Universität Wien  
**Fakultät für Informatik**

von

Dipl.-Ing. Lukas Gerhold  
Matrikelnummer: 9926672  
Mortaraplatz 11, 1200 Wien

Wien, im Mai 2011



## Deutsche Kurzfassung

Etwa zwei Prozent der österreichischen Bevölkerung leiden an schlecht heilenden oder chronischen Wunden. Die Patienten sind vor allem ältere Menschen und leiden oft an intensiven Schmerzen welche mitunter zu Immobilität und sozialer Isolation führen können. Die Behandlung von Patienten mit chronischen Wunden ist eine multi-professionelle Aufgabe die Ärzte und Pflegepersonal miteinschließt. Die Behandlung chronischer Wunden dauert meist über einen längeren Zeitraum und zielt darauf ab die Selbstständigkeit der Patienten zu erhalten.

Informations- und Kommunikationstechnologien (IKT) spielen eine wichtige Rolle zur Verbesserung der Qualität der Wundversorgung und Wundbehandlung, und zur Verbesserung der Koordination und Kommunikation im Rahmen des Wundmanagements. IKT Anwendungen ermöglichen eine objektive und einheitliche Wunddokumentation unabhängig von Ort und Zeit. IKT macht den Wundmanagementprozess nachvollziehbar und transparenter. Das Ziel diese Arbeit ist es zu zeigen wie Electronic Health Records (EHR) in einen Wundmanagementprozess eingebettet werden können um die Beziehungen zwischen Patienteninformationen und dem Behandlungsprozess darzustellen und nutzen zu können.

Einige ambitionierte Ansätze wurde kürzlich unternommen welche Modelle und Datenstrukturen für EHRs hervorbrachten und semantische Interoperabilität zwischen EHR Systemen ermöglichen sollen. Einer von ihnen ist der ISO 13606 EHR Kommunikationsstandard.

Die Web Ontology Language wurde entwickelt um Daten und Informationen semantisch zu beschreiben und zu verknüpfen. Die Web Ontology Language ermöglicht einen höheren Grad an Abstraktion und Expressivität als relationale Datenbankschemata.

In dieser Arbeit stelle ich das Ontology Driven Application Model vor, welches beschreibt wie man Applikationsverhalten dynamisch aus dem Wissen einer Ontologie interpretiert und ableitet. Des Weiteren beschreibe ich ein Prozessmodell das auf der Web Ontology Language beruht und ermöglicht die Prozessdefinitionen für den Wundmanagementprozess zu erstellen. Ich beschreibe ebenfalls die Umsetzung eines prototypischen Kommunikationsservers sowie Systeme die miteinander EHRs in einem Wundmanagementprozess austauschen.



## Abstract

**Context:** About two percent of the Austrian people suffer from bad healing or chronic wounds. The patients are mainly older people and often suffer from intensive pain that can cause immobility and social isolation. Treating patients with chronic wounds is a multi-professional task that involves specialists and nurses, often continues over an extended period of time and aims to obtain patients' self-management.

**Objectives:** To improve the quality of the wound management and wound care and to improve the coordination and communication of the wound management procedures information and communication technologies (ICT) play an important role. ICT applications facilitate establishing an objective and integrative wound-documentation independent of time and location and making the wound-management process traceable and more transparent. The aim of this work is to show a way how electronic health records can be embedded into the wound management process for providing relations between patient information and decision- and treatment process.

**Methods:** Recently ambitious attempts have been done defining structures and models for EHR facilitating semantic interoperability between electronic health records - and electronic patient record systems. One of them is the ISO 13606 EHR communication standard.

Recently the Web Ontology Language was developed for defining and structuring data semantically. The Web Ontology Language enables higher levels of abstraction and expressiveness for defining models than relational database schemata can.

**Results:** I introduce an ontology driven application model. The ontology driven application model describes how to derive functionality from ontology knowledge dynamically and using it for application behavior. Furthermore I introduce a process model based on the Web Ontology Language that facilitates the definition of a standardized and unambiguous process description for the wound management process. I also describe a prototype implementation of a communication server and two systems communicating EHRs.

**Conclusion:** ISO 13606 can be implemented using the ontology driven application model yielding all intensions the standard was developed for. These are for example: semantic interoperability, longitudinal data retrieval, etc.

Additionally processes can be dynamically interpreted through implementing the ontology driven application model. Following this approach the exchange of process definitions and process state information can be facilitated.



## Table of Contents

<b>1 Introduction.....</b>	<b>11</b>
1.1 Wound management.....	11
1.2 Evidence based medicine .....	12
1.3 The problem to be solved.....	14
1.4 The technical background.....	16
1.4.1 The integrated care electronic health record.....	16
1.4.2 Clinical guidelines.....	17
1.4.3 The integration of standalone tele-medical IT systems .....	18
1.4.4 The generic implementation method.....	19
<b>2 Standards and technologies.....</b>	<b>20</b>
2.1 ISO 13606 communication standard .....	20
2.1.1 The EHR component model (ISO 13606 part1).....	23
2.1.2 Archetypes (ISO 13606 part 2).....	29
2.2 The guideline interchange format .....	37
2.3 Semantic web- and ontology frameworks.....	38
2.4 Ontologies.....	41
2.4.1 Semantic web, resource description framework and web ontology language overview .....	44
2.5 OWL-S an ontology supporting process modeling and service interoperability .....	48
2.5.1 Service.owl:.....	48
2.5.2 Profile.owl:.....	49
2.5.3 Process.owl: .....	51
2.5.4 Grounding.owl .....	53
2.6 The development environment.....	55
2.6.1 The Glassfish application server and the J2EE standard .....	55
2.6.2 Web applications and Java ServerFaces .....	56
2.6.3 Service oriented design and web services.....	56
<b>3 Methods .....</b>	<b>60</b>
3.1 The GLIF methodology for defining electronic clinical guidelines.....	60
3.1.1 Action step .....	61
3.1.2 Decision step.....	62

3.1.3	Branch step.....	63
3.1.4	Synchronization step .....	63
3.1.5	Patient state step .....	63
3.2	Building semantic web applications.....	64
3.2.1	Architecture of a semantic web application .....	65
3.2.2	Ontology-driven software development.....	66
3.3	Ontology design and best practice.....	67
3.3.1	General design principals .....	67
3.3.2	How to design an ontology.....	70
3.4	Tools and application programming interfaces .....	76
3.4.1	Protégé – an ontology engineering toolset.....	76
3.4.2	Link EHR-ED archetype editor .....	77
3.4.3	OWL-S application programming interface.....	78
3.4.4	The NetBeans integrated development environment .....	79
<b>4</b>	<b>Results .....</b>	<b>81</b>
4.1	The idea .....	81
4.2	The wound management scenario.....	82
4.3	The basics of an ontology driven application.....	86
4.3.1	The ontology driven application model - ODAM.....	86
4.3.2	ISO 13606 and the ontology driven application model.....	89
4.3.3	Process representation for an ontology driven application model .....	89
4.4	The wound management process and the wound management archetypes .....	91
4.4.1	The GLIF compatible process-ontology .....	91
4.4.2	The wound management process .....	98
4.4.3	Wound management general patient info archetype .....	102
4.4.4	Wound management archetype – wound specific .....	104
4.4.5	Wound management prescription archetype – wound specific	106
4.4.6	The wound management treatment archetype – wound specific .....	108
4.5	System architecture .....	110
4.5.1	Ontologies and data repositories .....	111
4.5.2	Communication server and web services .....	112



4.5.3	Communicating systems .....	116
4.5.4	Ontologies .....	120
<b>5</b>	<b>Discussion and related work .....</b>	<b>125</b>
5.1	Wound management .....	125
5.2	Telemedical wound care .....	127
5.3	Ontologies and ISO 13606 .....	129
5.4	Ontologies and clinical guidelines .....	131
5.5	Interoperability in health care data exchange .....	134
<b>6</b>	<b>Conclusion and Perspectives .....</b>	<b>140</b>
<b>7</b>	<b>Acknowledgements .....</b>	<b>143</b>
<b>8</b>	<b>Appendix .....</b>	<b>144</b>
8.1	Archetypes in ADL format.....	144
8.1.1	Wound management general patient info archetype.....	144
8.1.2	Wound management archetype – wound specific.....	148
8.1.3	Wound management prescription archetype .....	156
<b>9</b>	<b>References.....</b>	<b>160</b>



# **1 Introduction**

This section is divided into four subsections. First I will describe the medical idea behind the practical project. Subsequently I will describe the meaning of evidence based medicine then I will describe the problem to be solved. Finally I will describe the technical background of this work.

## **1.1 Wound management**

The idea for this work came up because of the medical need for a tele-dermatologic monitoring-system for patients with chronic wounds. Content of this section is the medical need that lead to the thesis.

About two percent of the Austrian population i.e. around 161.000 persons suffer from bad healing or not healing wounds (1). The affected persons are mainly older people, in 2001 more than 160.000 Austrians were older than 60 years (2). As people tend to get older the number of patients with chronic wounds is expected to rise. Treating patients with chronic wounds often continues over an extended period of time and often lasts for month and years. High costs are the consequence (2).

Treating chronic wounds is a multi-professional task that aims to obtain patients' self-management and to improve patients' wellbeing (3). The patients often suffer from intensive pain that can cause immobility and even social isolation. Many studies provide evidence that a lot of patients often feel that physicians and nurses do not look at the patients' problems in their entirety and focus on the patients' wounds only (3, 4).

Providing high-quality wound-management, multi-professional teams (physicians and nurses) have to closely work together. Physicians and nurses need extraordi-

nary communication skills to provide best applicable care for each patient's individual care problem (4). A lack of well-educated health care professionals and communication problems delay and exacerbate the healing process and decline the quality of wound care. That is why the teamwork of multidisciplinary and multi-professional teams and the development of an objective wound-documentation have to be promoted and supported by any means (4).

To improve the quality of the wound management and wound care, information and communication technologies (ICT) play an important role. ICT applications facilitate to implement an objective and integrative wound-documentation. Especially telemedical applications facilitate patients' wound screening independently of time and location (4).

Telemedical support is of great importance when it comes to establish a multi-professional homecare-management system. Through ICT, physicians do not need to visit their patients physically, but may still provide their consultations. This results in higher efficiency for the attending physicians and the advantage of homecare and treatment for the patients. Chronic wound patients are often confronted with long latencies for their necessary medical checkups, frequently affiliated with long journeys to reach their attending physician. Especially the medical care of immobile patients in areas with few specialists, telemedicine could help to provide a fast and easy way of chronic wound treatment.

## **1.2 Evidence based medicine**

Evidence based medicine (EBM) aims to provide the best available evidence for medical decision making gained from scientific methods (5). It aims to assess the strength of evidence of the benefits and risks of treatments and diagnostic tests (6). EBM offers the most objective and most reliable way to maintain and assess

consistently high quality and safety standards in medicine. EBM facilitates speeding up the process of transferring clinical research results into practice and has the potential to reduce health-care costs (7).

The Cochrane Collaboration plays an important role in the EBM movement (8). Its intended use is to provide up-to-date systematic reviews of randomized controlled trials from all areas of health care to make the best available evidence accessible for the purpose of healthcare decision making (7, 8). The major outcome of the Cochrane Collaboration is the Cochrane Library. The Cochrane Library comprises (7, 8):

- a database of systematic reviews,
- a database of abstracts of reviews of effectiveness (DARE),
- a methodology register,
- a controlled trials register,
- a health technology assessment database,
- and a National Health Service (NHS) economic evaluation database.

EBM can be systematically applied in practice by defining guidelines and checklists which represent the medical knowledge. Through these representations physicians are guided through the decision or treatment process (9). Their content is based on sources for evidence based care and systematic reviews. The aims of clinical practice guidelines are (9):

- describing appropriate medical care based on the best available scientific evidence and broad consensus,
- eliminating inappropriate variations in practice care,
- providing - best possible - rational basis for referral,
- promoting efficient use of resources,

- building the focus of quality control, including audits,
- highlighting shortcomings of existing literature and suggesting future research.

Information and communication technologies (ICT) can support providing EBM. Computerized electronic guidelines can automatically generate EBM recommendations about what actions to perform according to a patient's health status (9). In many ways electronic guidelines provide more benefits than paper-based guidelines (9):

- they facilitate improving the clarity of a guideline, e.g. in decision criteria or clinical recommendation,
- they are offering a (better) description and overview of a patient state,
- they can automatically calculate timely, patient specific decision support, warnings, and reminders

### **1.3 The problem to be solved**

A telemedical monitoring system that optimally supports physicians and nurses performing a modern wound management must support the communication of patient information. Furthermore, wound management defines integrated tasks involving physicians and nurses to closely work together to find the optimal treatment of the patients' needs (4). The legislative defines a minimum set of tasks as well as the responsibilities between physicians and nurses to assure a basic level of quality of treatment. Additionally evidence based medicine defines tasks and processes to refine the legally ones for optimally diagnosing the patients wounds and treating the patient (4).

A modern home care treatment of patients with chronic wounds has to fulfill the legal rules which are that the diagnosis and the treatment have to be document-

ed and performed by the physician. And further, the responsibility of the nurse is to document the patient's health status and to perform the treatment according to the physician's prescription (4).

The real world scenario I am addressing in this work is that the nurse is located at the patient's site to treat the patient at home. The nurse is an employee at any home care organization. The physician who interacts with the nurse is any of the following: a general practitioner or a specialist with an own practice, or a physician at the hospital. The physician is not located at the patient's site. Independent where the physician or the nurse is located or to which organization they belong, for treating a patient the physician and the nurse have to work together to treat a patient with chronic wounds. Exchanging patient related information is necessary to deliver the right information to perform a certain task, like diagnosing or treating the patient. It is also necessary to define the procedure to evaluate which step is next when treating a patient. Within an organization like a hospital where physicians and nurses closely work together, the communication procedure is already defined - often inclusively as physicians and nurses know their responsibilities and the medical documentation provides them with the necessary patient related information. But when it comes to a communication between institutions the communication procedure including the information to be exchanged is not yet defended but has to be defined exclusively and transparently.

The information problem I am addressing in this work is that services (like diagnosing, prescribing a treatment, or treating a patient) of different institutions and of different professions (physicians and nurses) which exchange patient related information have to be embedded in a communication process in a way that any involved system can evaluate which task or service has to be performed next to treat a patient with chronic wounds. In other words if a system provides stand-

ardized patient related information it is a priority not clear for which purpose this information has been captured and what to do with this patient related information next. Blobel mentions this problem in (10) and names its solution as institutional or service interoperability.

## **1.4 The technical background**

In the following the central technical components are described which build the foundation of this work.

### **1.4.1 The integrated care electronic health record**

Integrated care is the final aim for the development of an electronic health record (EHR). Integrated care means: health delivery through multi-specialty and multi-disciplinary teams, sometimes called shared care or co-ordinated care. Especially for chronic disease and episodic or periodic conditions, integrated and shared care fits very well (11).

Integrated care is usually planned and delivered over an extended period of time, particularly for the management of chronic diseases. This introduces the notion of a longitudinal record, with information recorded about past, present, and future events and plans. The integrated care EHR definition is based on these characteristics (11).

The International Organization for Standardization (ISO) defines an EHR for integrated care as:

“A repository of information regarding the health status of a subject of care in computer processable form, stored and transmitted securely, and accessible by multiple authorized users. It has a standardized or commonly agreed logical in-



formation model which is independent of EHR systems. Its primary purpose is the support of continuing, efficient and quality integrated health care and it contains information which is retrospective, concurrent, and prospective” (11).

To optimally support multidisciplinary health care teams, evidence based medicine, good clinical practice and health care management facilitating to standardize the health care processes and support interdisciplinary communication, is of great importance. Process orientation and process interoperability play a crucial role when it comes to realize integrated care based on EHR (10).

#### **1.4.2 Clinical guidelines**

Clinical guidelines are statements systematically developed to support general practitioners in making clinical decisions and managing medical actions more effectively (12). Clinical guidelines are usually plans and procedures for treatment and are often used for developing the clinical decision support system (13). The aim of clinical guidelines is to reduce interpractice variations and costs of medical services, improve the quality of care, and standardize clinical procedures (14). A wider adoption of computerized clinical practice guidelines, however is yet to be realized (13). The reason for that can be stated as “The failure of integration of guideline implementations with clinical workflows” (15, 16).

Many computer interpretable models have been developed for modeling clinical guidelines like the GuidLine Interchange Format (GLIF) (17), ASBRU (18), ARDEN (19) and EON (20). The guideline execution engine (GLEE) (21), guideline acquisition, representation and execution (GLARE) (22), and digital electronic guideline library (DeGeL) (23) have been developed to demonstrate that guideline definitions can be executed automatically to support the decision making process. “However, these execution engines often address the automation in a single homogeneous healthcare organization, and custom adaptation phases are required

to communicate with clinical applications such as for accessing the patient records or invoking medical services” (13). In the GLIF specification (24) the lack of integration support is stated as: “there is a need for an implementable specification that can be incorporated into an institutional system where the actions specified must be mapped to institutional procedures, and the patient data references must be mapped to the electronic medical records of the underlying system.”

In this work I will utilize the knowledge behind the GLIF model that is used for formulating clinical procedures and processes. The expressivity of the GLIF model in combination with the OWL-S process model (see section 2.5) facilitates to define a process that semantically describes services that access standardized EHR. This semantic description facilitates to automatically interpret the state of the communication process during the telemedical wound treatment involving physicians and nurses at different site and locations.

### **1.4.3 The integration of standalone tele-medical IT systems**

Many heterogeneous IT systems have been developed serving (tele-) medical needs and supporting medical documentation throughout proprietary data handling. Many of these applications are standalone applications for certain medical purposes that are not integrated in hospital or medical office information systems (25). A lot of effort has to be done to integrate these applications into hospital information systems, medical office information systems or local electronic patient record systems. Unfortunately, many of these systems are facing a lack of acceptance although these systems seem to fulfill their purposes. The reasons for the lack of acceptance are not completely inquired yet (25). However, a well-functioning integration of telemedicine applications with hospital information systems, medical office information systems or local electronic patient records

making data entry more efficient and patient information available independent of localization and time, will play an essential role for improving user acceptance. The ISO 13606 EHR communication standard was developed for exchanging patient related health information beyond the borders of local medical information systems (26, 27). This work shall be a contribution to show how an implementation of ISO 13606 can look like and how to practically use ISO 13606 interfaces for EHR communication.

#### **1.4.4 The generic implementation method**

The ISO 13606 EHR communication standard is based on the dual model approach. The dual model approach is going to be explained in more detail in the section 2.1. However to explain the motivation, I have to anticipating some aspects of the 13606 standard and ontologies. The concept of the dual model approach has its origin in (28, 29) that dealt with ontologies for knowledge representation and knowledge sharing. I reflected upon these works and combined them with new approaches of knowledge representation and sharing that lead to the development of the web ontology language (OWL) (30). The motivation inspiring me was to find a method how to generically use and implement the power of ontologies for knowledge representation and knowledge sharing. This method shall provide a basic approach for vendors, how to implement the 13606 standard for their (legacy) systems.

## 2 Standards and technologies

In this section I will provide an overview about the standards and technologies used in this work. There are many standards that deal with the topic of an EHR. As this work focuses on the ISO 13606 standard some of the other standards are only mentioned here to show that although standard development is in progress, there are still heterogeneous definitions and even different standards applicable. There are efforts to harmonize these standards and to achieve interoperability between these standards (31, 32). I would like to mention the most important ones only:

- Health Level Seven (HL7) (33, 34),
- GEHR/openEHR (35-37),
- ISO 13606 Communication Standard (26, 27, 38),

Others that are not explicitly developed for EHR but are also mentioned in literature being applicable for implementing EHR-systems (39):

- Web-Services, and Service Oriented Architecture (SOA) (40-42),
- Model Driven Architecture, and Common Object Request Broker Architecture (CORBA) (43-45).

This work focuses on the ISO 13606 communication standard. Further details or comparisons about other standards are not focus of this work.

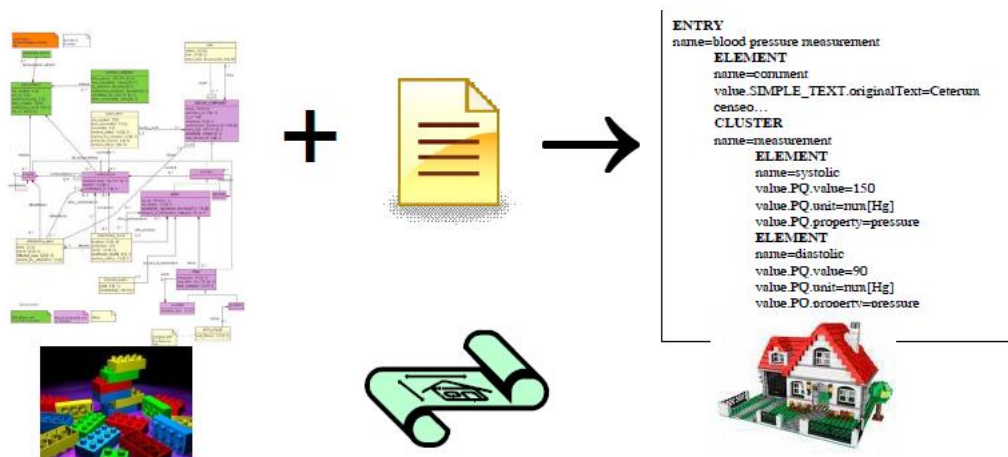
### 2.1 ISO 13606 communication standard

First this section will provide a brief overview of the ISO 13606 standard. Subsequently I am going to describe the 13606 reference model and archetype model in more detail.

ISO 13606 is based on the GEHR/openEHR dual model approach. The dual model approach consists of an archetype model that constraints and references a generic reference model (36). In other words an instance of the archetype model can be seen as description how to use the reference model - through constraining and referencing it.

The following paragraph taken from (36) describes and illustrates the **dual model** approach in an easy way:

“One way to understand archetypes is to imagine that the Reference Model (RM) defines the engineering specification of LEGO® bricks from which, as every child, and not a few adults know, anything can be built. The semantics of the RM are analogous to the semantics of Lego bricks, i.e. the engineering specification of the particular coupling and joining mechanisms built into the bricks. The set of all possible combinations of a particular set of bricks comprises a vast construction space. However, most combinations are meaningless - only a tiny proportion of the space consists of the interesting constructions of houses, dogs, and tractors; all other combinations are legal if the bricks are connected correctly, but have no meaning to us, the users. Likewise, a RM defines a vast informational construction space, only a small proportion of which contains combinations valid in the domain. Consider further that the valid Lego brick constructions cannot be divined from the bricks themselves: they come from fertile imaginations, or else printed plans included in Lego packages. It is often the case that small variations and optional add-ons are suggested for the one model; this means that the set of all possible variants on the model form a constellation of brick combinations corresponding to the one plan, or model definition. Such plans are the Lego versions of archetypes”. The LEGO® metaphor is shown in Figure 1.



**Figure 1: Lego analogy, from left to right:  
reference model, archetype model, integrative archetype (46)**

Now as we know the design paradigm of the ISO 13606 communication standard, I would like to introduce the 5 parts of the ISO 13606 standard.

- Part 1 specifies the reference model, which is a generic information model for communicating patient information.
- Part 2 specifies the archetype model, which is also a generic information model and language to specify the aforementioned archetypes for communicating domain concepts.
- Part 3 specifies the reference archetypes and terms lists. It describes how other EHR standards could comply with 13606.
- Part 4 specifies the security features.
- Part 5 specifies the interface model which describes how an interoperable communication can take place.

The following sections describe in more detail the contents of each part. The descriptions are taken from the official ISO Webpage:

### 2.1.1 The EHR component model (ISO 13606 part1)

The goal of ISO 13606 standard is to provide a data structure that holds the information of a patient. The ISO 13606 was designed to formulate EHR extracts and communicate these extracts semantically interoperable. Model definitions (archetypes) and data- (EHR extracts) are both being communicated, thus a receiver system is able to interpret the data without complicate interfaces definitions on database and item level.

The following figures (Figure 2 and Figure 3) illustrate the structure of an EHR extract.

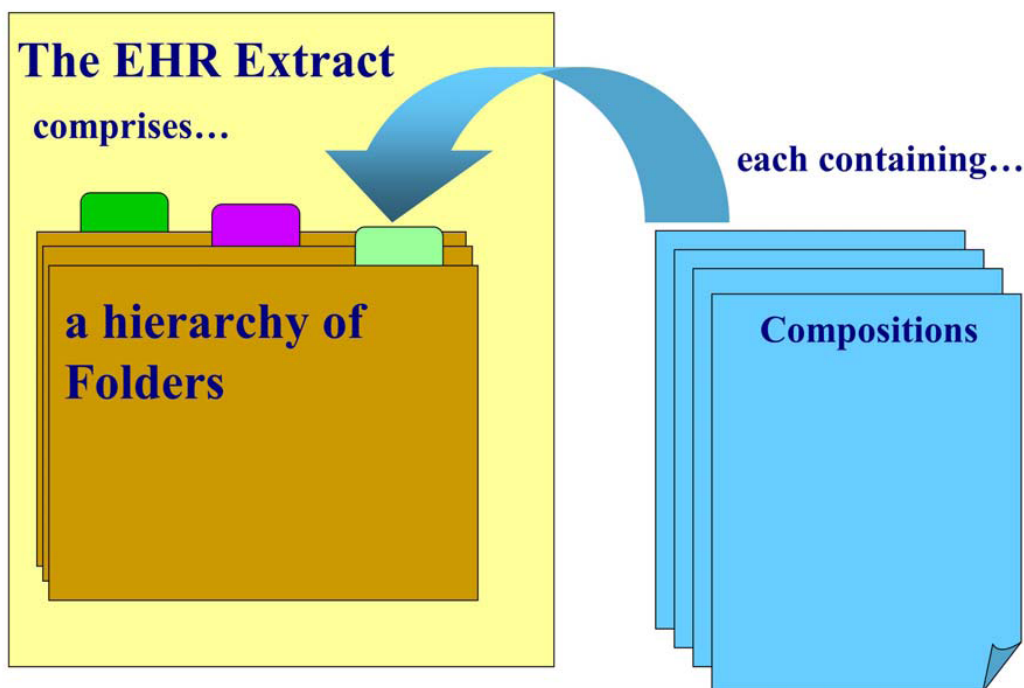
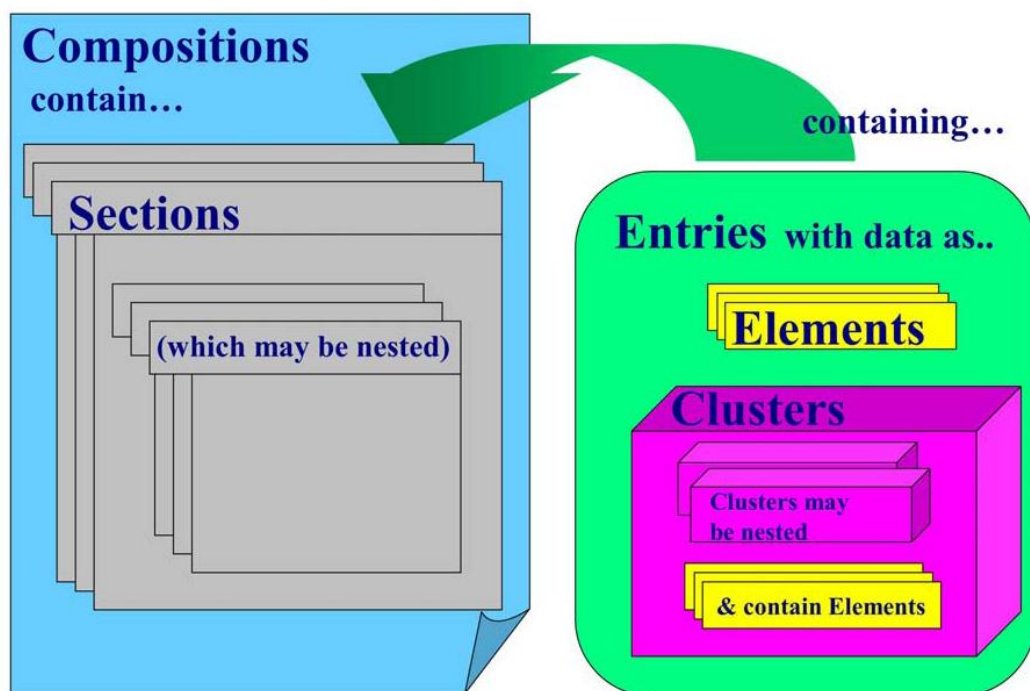


Figure 2: EHR Extract hierarchy (part 1) (26)

Figure 2 shows the top level structures of an EHR extract and how it is arranged. EHR Extract, Folder and Composition shown in Figure 2 are classes of the reference model.

Section, Entry, Cluster and Elements shown in Figure 3 are also classes of the reference model. Figure 3 illustrates the deeper levels of an EHR extract, starting with the Composition (compare Figure 2).

Figure 2 and Figure 3 shall provide the reader with an overview of an EHR extract and how the instances of the reference model classes are arranged to express clinical information.



**Figure 3: EHR Extract hierarchy (part 2) (26)**

The reference model comprises 4 packages of classes:

1. Extract package (illustrated in Figure 4)

The extract package comprises the main reference model classes which are used by the archetype model:

- EHR\_EXTRACT



- RECORD\_COMPONENT
- FOLDER
- COMPOSITION
- SECTION
- ENTRY
- ITEM, CLUSTER and ELEMENT

and other principal classes of the reference model:

- AUDIT\_INFO
- FUNCTIONAL\_ROLE
- ATTESTATION\_INFO
- RELATED\_PARTY
- LINK

## 2. Demographic package

The Demographic package comprises all classes necessary to describe demographic data of patients, subjects and any person who participated in the health care process. “The goal of this part of the model is to provide a necessary and sufficient description of each entity to support human interpretation of the EHR, and demographic matching to enable the EHR Recipient to identify the corresponding entities within its own demographic server”(26).

The whole DEMOGRAPHIC\_EXTRACT is optional and need not to be shared, e.g. if both communication parties (EHR provider and EHR recipient) share the same demographic server.

For further details please refer to (26).

### 3. Support package

The support package comprises classes for complex data types. These classes are referenced in the Extract package.

### 4. Primitives package

This package comprises classes that describe the primitive types, like Array, Boolean, Integer, String, etc.

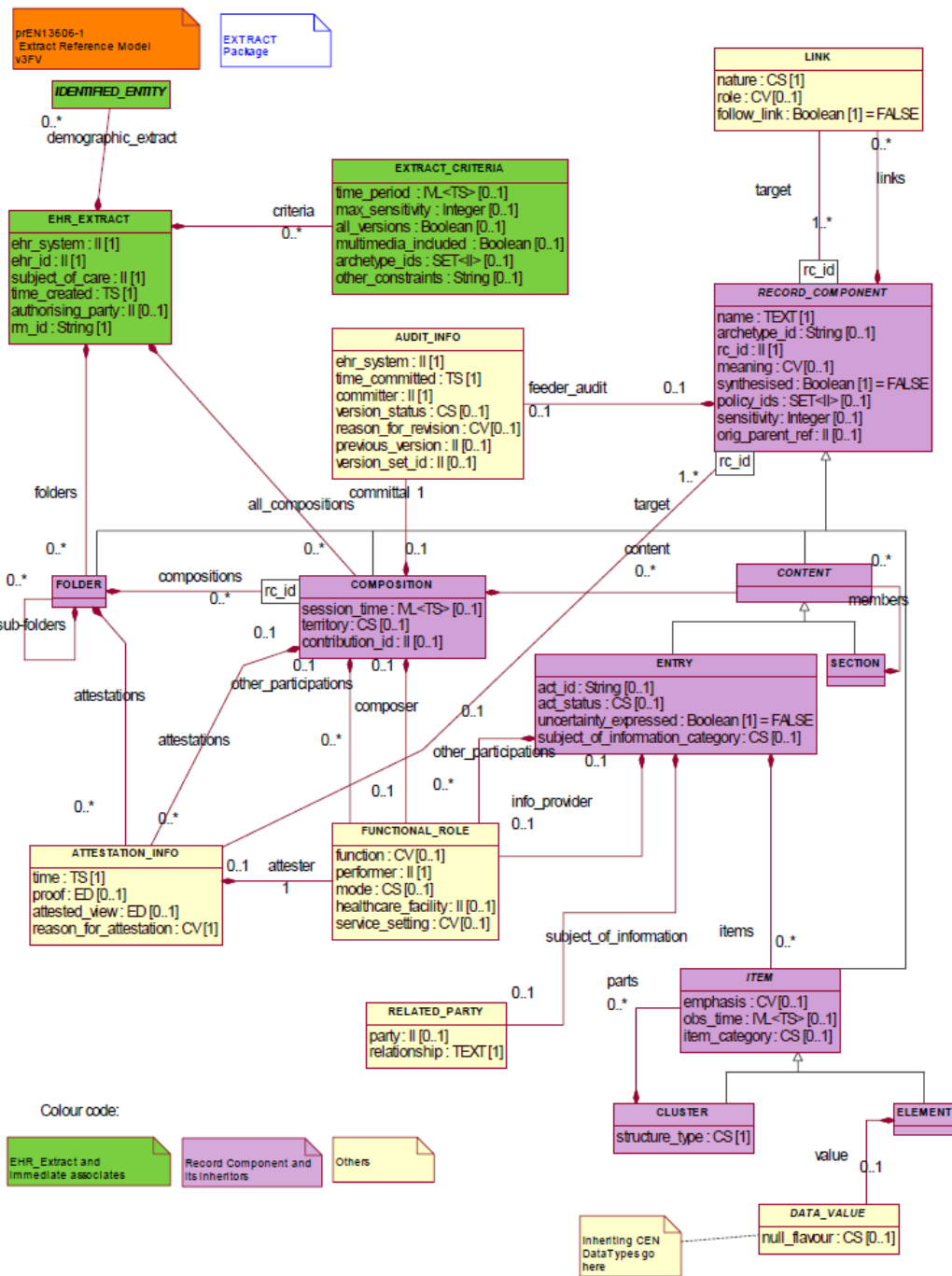


Figure 4: Extract Package (26)

Distributed electronic health records will be implemented by proprietary interfaces. Each system that integrates electronic health records has to implement its own proprietary interface. Medical data from medical applications like hospital information systems, databases or medical devices have to map their internal data to and from the electronic health record. The electronic health record is used as a generic data format to exchange the data between the participating systems. Messages and interfaces provide the technical layer for communicating the electronic health records.

The generic reference model of ISO 13606 is designed for modeling all kinds of data and data structures. An EHR Extract which is part of the ISO 13606 reference model is intended to be exchanged between the communicating systems and devices. Even if the communicating systems and devices have not agreed in advance on the information structure of an EHR extract, an EHR extract comprises sufficient information to be interpreted by the communicating parties. This information comprises labels, structures, context necessary information and of course the medical information.

The ISO 13606 reference model is specified as an Open Distributed Processing (ODP) Information Viewpoint Model. “The RM-ODP family of recommendations and international standards defines essential concepts necessary to specify open distributed processing systems from five prescribed viewpoints and provides a well-developed framework for the structuring of specifications for large-scale, distributed systems” (47). One of these viewpoints is the Information Viewpoint mentioned above. It focuses on the semantics of the information and the information processing performed. The Information Viewpoint describes the information handled and processed by the system and the structure and content type of the supporting data (48). The 13606 reference model comprises a framework of generic classes (see section 2.1.1) that represent the building blocks of an EHR.

It represents the stable characteristics of an EHR and would be communicated by a distributed EHR environment using specific messages or interfaces (defined in ISO 13606 part 5).

### **2.1.2 Archetypes (ISO 13606 part 2)**

Furthermore the sharing of EHRs and their meaningful interpretation across distributed sites requires a consistent approach being used for clinical data structuring. These clinical data structures represent the semantics of a clinical context and are used to communicate via the reference model. The definition of the semantic data structure is used to represent equivalent clinical information consistently. This enables clinical applications or analysis tools to safely process EHR data that have come from diverse heterogeneous systems (27).

To reach the goal of EHR interoperability we need a generalized approach to represent every conceivable kind of health record structure in a consistent way. This approach has to meet the requirements of an EHR defined by any profession, specialty or service. Data sets, value sets, templates etc. required by different health care domains will be diverse, complex and will change frequently as clinical knowledge and medical practice advance.

The field of semantic interoperability in health informatics aims to cope with these requirements.

Archetypes are pre-coordinated definitions of RECORD\_COMPONENT hierarchies. Archetypes are agreed within a community to ensure data consistency, data quality and semantic interoperability in order to electronically communicate.

Within an EHR\_EXTRACT the patient information is structured by a hierarchy of RECORD\_COMPONENT subclasses. An archetype specifies (constrains) a particu-

lar hierarchy of RECORD\_COMPONENT subclasses. Furthermore an archetype defines or constrains the names of the RECORD\_COMPONENT subclasses, their data types, and other relevant attribute values as well as optionality and multiplicity in the hierarchy. An archetype also constrains the data types and value ranges of ELEMENT which is the leaf class of the RECORD\_COMPONENT subclasses. Archetypes themselves are instances form a formal model, the archetype model. It is a constraint model and it is also specified as an ODP Informational Viewpoint Model (described above). The archetype model itself is stable. Individual archetype model instances (archetypes) however, can be revised, refined or succeeded by others as clinical practice and medical knowledge evolves. Version control ensures that EHR\_EXTRACT data which are constrained by previous versions of an archetype do not become invalidated.

Archetypes can be used within EHR systems to manage the EHR data committed to a repository. However as ISO 13606 claims to be an interoperability standard, it does not state any assumption of how archetypes are used within an EHR provider system if ISO 13606 is used for EHR communication. ISO 13606 assumes that the original EHR data from a provider system may be assigned to a set of archetypes when creating the EHR\_EXTRACT (with that set of archetypes and EHR data).

Some attributes of the reference model defined in ISO 13606 part 1 can be used within an archetype to specify any structure of a RECORD\_COMPONENT of an EHR\_EXTRACT. The class RECORD\_COMPONENT has an attribute *archetype\_id* that identifies the archetype and its nodes. The *meaning* attribute of the RECORD\_COMPONENT class refers to the concept represented by the corresponding archetype node, if the RECORD\_COMPONENT is constrained by an archetype. It should be mentioned that ISO 13606 part 1 does not require to use archetypes for managing RECORD\_COMPONENTs and its hierarchies within an

EHR\_EXTRACT. The reference model defines the archetype-related attributes as optional (27).

### 2.1.2.1 Overview of the archetype model

This section provides an informative overview of the archetype model.

The archetype model consists of classes and attributes for identifying information of an archetype, a *description* (its metadata), a *definition* (expressed by constraints on instances of an object model), and an *ontology*. The ARCHETYPE class defines the attributes for identifying information and the lifecycle state. The ARCHETYPE\_DESCRIPTION package defines classes for revision history (attribute *revision\_history*) information, and for descriptive information (attribute *description*) about the archetype. The descriptive information describes the archetype itself and provides the metadata. The revision history information provides information about the committal of the archetype to a repository. It takes the form of a list of audit trail items.

The *definition* attribute in the ARCHETYPE class refers to the C\_COMPLEX\_OBJECT class and can be considered as the main part of an archetype. An instance of the C\_COMPLEX\_OBJECT builds the root of the constraint structure of an archetype and shall always take the form of a constraint on a non-primitive object type. The last main attribute within the ARCHETYPE class I want to mention is the *ontology* attribute. It refers to the ONTOLOGY class and allows an archetype to be natural-language and terminology neutral. See Figure 5 and section 2.1.2.2 for more details.

Within the archetype package, there is also an enumeration class VALIDITY\_KIND included. Its intended use is for attributes within the constrained model which refer to the VALIDITY\_KIND class and whose values can become “mandatory”,

“optional”, or “disallowed”. Such attributes are part of the classes C\_DATE, C\_TIME and C\_DATE\_TIME. Within these classes the VALIDITY\_KIND typed attributes are used to constrain time options. For more details refer to (27). The VALIDITY\_KIND class is not illustrated in Figure 5.

The ARCHETYPE class defines the *original\_language* attribute which records the original language of an archetype. This attribute is important to describe the natural language elements of an archetype which are the description and the ontology definitions. For details how to translate an archetype refer to (27).

#### **2.1.2.2 The archetype definition**

As mentioned before the purpose of an archetype is to define and to constrain the hierarchy levels of RECORD\_COMPONENTs within an EHR\_EXTRACT. These definitions are made within the definition part of an archetype. This part consists of alternate layers of object- and attribute-constraining nodes, each constraining the next level of nodes. An “attribute” is any data property; it may be an association, aggregation, composition or a primitive attribute like a value. The layers of object- and attribute-constraining nodes span a tree with a root element which is an instance of C\_COMPLEX\_OBJECT. At the leaves there are primitive object constraining nodes for primitive types like String, Integer, etc. Internal references, like constraint reference nodes referring to text constraints in the constraint binding part of the archetype ontology, and archetype constraint nodes representing constraints on other archetypes included in the current one, are also represented by nodes.

The most important classes (and nodes) of the constraint\_model package shown in Figure 5 are:



- `C_COMPLEX_OBJECT`: a class the instances of which represent nodes that constrain instances (objects) of non-primitive types, e.g. `ENTRY`, `SECTION`;
- `C_ATTRIBUTE`: the instances of this class represent constraints on attributes of objects (i.e. UML “relationship” or “primitive attribute”);
- `C_PRIMITIVE_OBJECT`: the instances of this class represent nodes for constraining primitive object types;
- `ARCHETYPE_INTERNAL_REF`: these instances represent nodes which refer to a previously defined object nodes in the same archetype; the reference is made using a path;
- `CONSTRAINT_REF`: instances of the `CONSTRAINT_REF` class represent nodes which refer to constraints on (usually) text or coded term entities. These entities appear in the ontology section of the archetype, and in ADL and are referred to using “acNNNN” code. The constraints are expressed in terms of queries on the external entity which may be an ontology or terminology.
- `ARCHETYPE_SLOT`: its instances represent nodes which define which other archetypes may be used at those points in the current archetype. The semantic is equal to the semantics of `C_COMPLEX_OBJECT` except that the constraints are expressed in another archetype and not the current one (27).

Figure 5 shows the archetype package and its comprising packages.

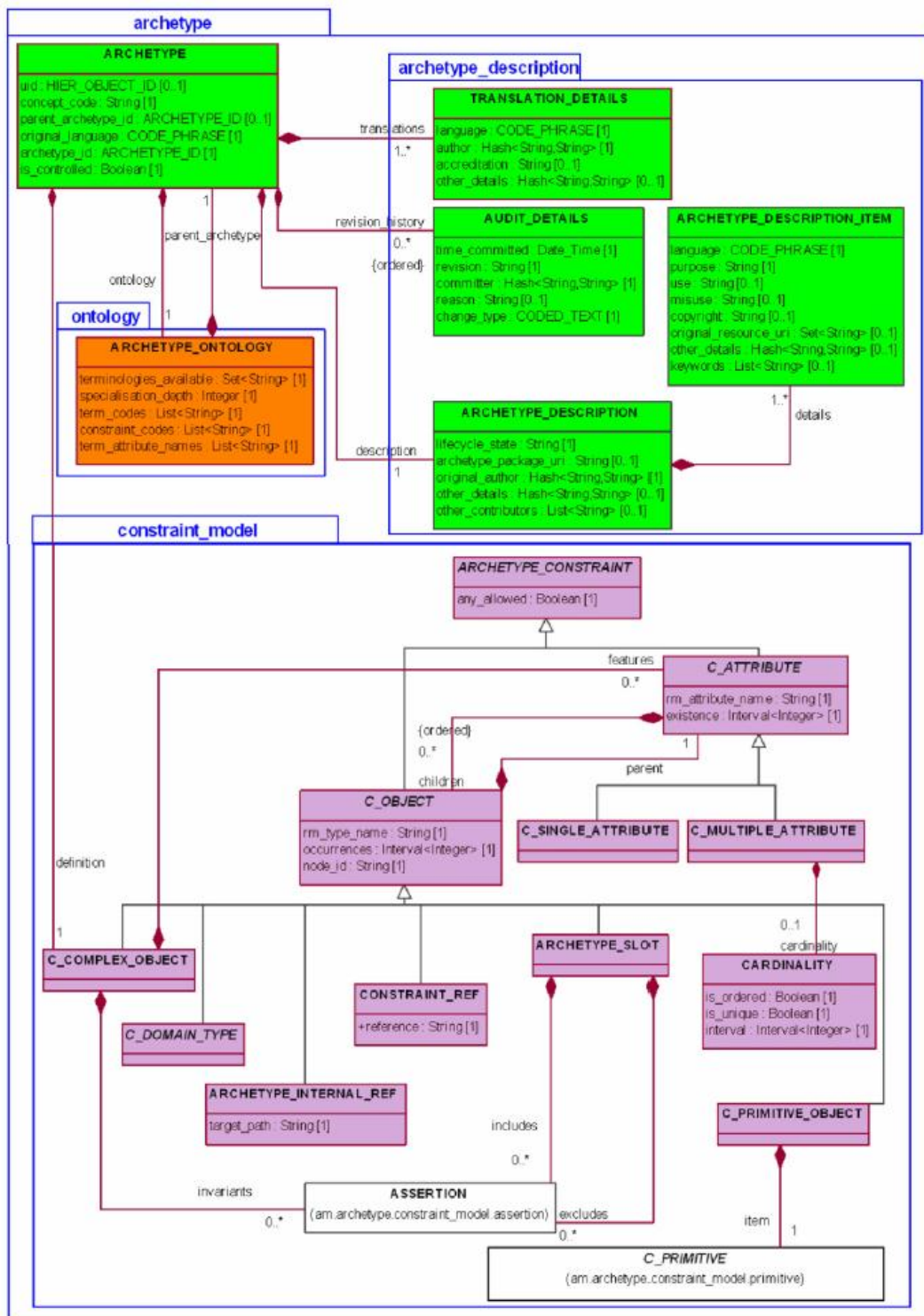


Figure 5: Overview of the main part of the archetype model (27)

### 2.1.2.3 Overview of archetype definition language

Before I show an example of an archetype see next section (section 2.1.2.4), I would like to provide an overview of the Archetype Definition Language (ADL), which is a formal language for expressing archetypes. An archetype expressed in ADL can be seen as a file written in a programming language (ADL) with a defined syntax. ADL comprises two different syntaxes, dADL and cADL. dADL syntax is used to express data that appears in the language, description, ontology, and revision\_history sections of an archetype written in ADL. cADL syntax is used to code the main part of an ADL archetype, the archetype definition section. The structure of an archetype written in ADL is shown in Figure 6 (27).

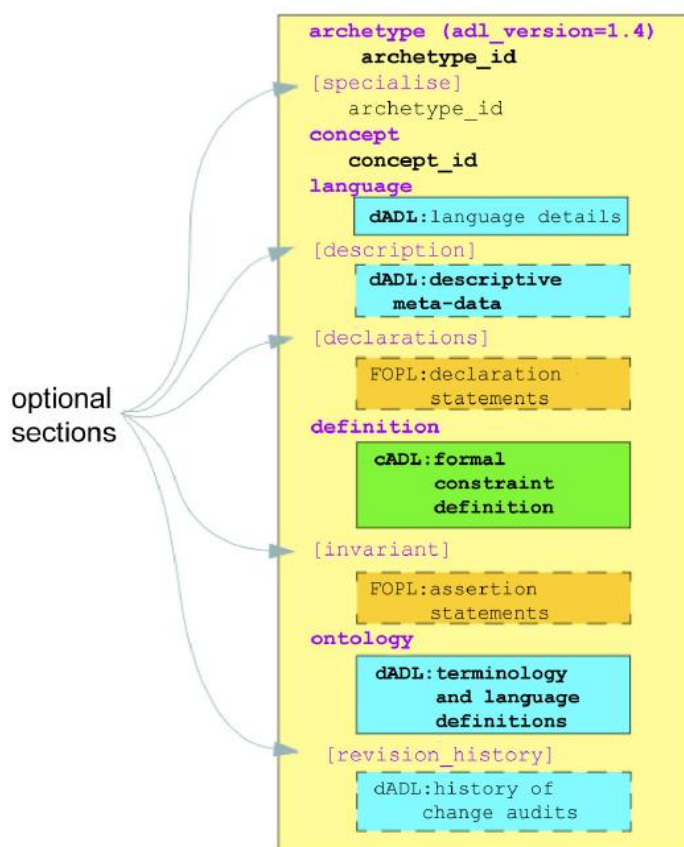


Figure 6: ADL archetype structure (27)

#### 2.1.2.4 Example of an archetype written in ADL

Figure 7 shows an example of a simple archetype. This archetype defines the concept of a *guitar* constraining a generic model of INSTRUMENT. The green colored names within the definition section (INSTRUMENT, size, date\_of\_manufacture, PART, etc.) define the classes and attributes (of the INSTRUMENT model) onto which the constraints will take their effects. Each block of braces defines the constraints on a particular set of instances of a class model. Each block can be seen as the specification of a concept such as guitar or neck. Blocks can be nested. As described above these nested blocks are spanning a tree and at the leafs of this tree there are the constraints on primitive types like Integer, String or Boolean (27).

```

archetype (adl_version=1.4)
  adl-test-instrument.guitar.draft
concept
  [at0000] -- guitar
language
  original_language = <"en">
  translations = <"de", ...>
definition
  INSTRUMENT[at0000] matches {
    size matches {60..120} -- size in cm
    date_of_manufacture matches {yyyy-mm-??}
    -- year & month ok
    parts cardinality matches {0..*} matches {
      PART[at0001] matches { -- neck
        material matches {[local::at0003]} -- timber
      }
      PART[at0002] matches { -- body
        material matches {[local::at0003]} -- timber
      }
    }
  }
}
ontology
  term_definitions = <
    [en] = <
      items = <
        ["at0000"] = <
          text = <"guitar">;
          description = <"stringed instrument">
        >
        ["at0001"] = <
          text = <"neck">;
          description = <"neck of guitar">
        >
        ["at0002"] = <
          text = <"timber">;
          description = <"straight, seasoned timber">
        >
        ["at0003"] = <
          text = <"nickel alloy">;
          description = <"frets">
        >
      >
    >
  >

```

Figure 7: archetype example (27)

## 2.2 The guideline interchange format

The guideline interchange format (GLIF) is a formal language that can be interpreted by computers. GLIF aims to model clinical practice guidelines to execute them electronically (49). GLIF specifies a model for representing shareable computer-interpretable guidelines. GLIF3, which is the current version of GLIF, covers

substantial updates and enhancements of the previous model introduced with GLIF2. GLIF3 facilitates encoding guidelines at three levels of abstraction:

1. A flowchart of actions, providing an overview on the guideline and modeling the conceptual part of the guideline. The flowchart represents the highest degree of abstraction.
2. A computable specification that can be verified for logical consistency and completeness.
3. An implementable specification, representing the most concrete representation which is intended to be incorporated into specific institutional information systems.

Implementations of GLIF3 have been evaluated on a wide variety of guidelines which are typical for the range of guidelines in clinical use. GLIF3 builds upon GLIF2 and adds various constructs to enable automatic interpretation and electronic decision-support systems. In order to facilitate integration of guidelines with clinical information systems, GLIF3 promotes standards being developed in Health Level 7. The GLIF3 specification comprises an extensible object-oriented model for specifying a guideline, and a structured syntax based on the resource description framework (RDF). A validation of GLIF3's ability to generate appropriate recommendations has been tested empirically by executing encoded guidelines against actual patient data (17, 50).

Please see section 3.1 and 4.4.1 for further details how GLIF relates to my work.

### **2.3 Semantic web- and ontology frameworks**

There are several ontology engineering tools like Protégé or other ontology editors for creating ontologies. These tools are often expert-oriented and designed for ontology engineers. The resulting ontologies of these tools have to be repre-

sented, refined and prepared to meet the requirements of an intuitive, user friendly and interactive application design especially for clinical use. Furthermore these clinical applications manipulate or even design ontologies themselves, dependent on their application design. In this section I will explain the technologies that facilitate ontology engineering. Ontology engineering comprises the disciplines of developing ontologies with tool support. Ontology engineering also deals with application programming interfaces (APIs) that help application programmers to access and manipulate ontologies.

The following introduction and overview of ontology frameworks and ontology engineering tools are taken - and slightly adapted - from the book "Ontologies-Based Business Integration" (51).

Ontologies can exist in form of files. For example a common format for OWL ontologies is XML and their underlying representation are XML files. For implementing ontology-driven applications ontologies have to be represented in the programming language itself. Thus a programming language is able to access, interpret or manipulate the ontology data. Frameworks and application programming interfaces (API) can provide the aforementioned representation. Object oriented programming languages e.g. represent ontologies and ontology data in form of classes and instances (52). Furthermore frameworks and APIs enable efficient development of modular, extendable, reusable and high-quality software implementations based on semantic web technologies (53).

There are many frameworks for working with ontologies. These frameworks differ in their sets of features (like scalability, expandability, robustness, database support, etc.), availability of documentation and support (54-56). There are open source implementations as well as commercial ones. The most common implementations are developed in Java (55).

Basic ontology framework features are common to almost all frameworks, even if they have different architectures and a different set of features.

A framework consists of a data model for processing ontology data. If we are talking about object oriented programming languages this model is called object model. Types of ontology information (concepts, instances, properties) are modelled as classes in this object model. The classes of the object model have methods for reading ontology data (e.g. by listing all properties of a concept), and for manipulating an ontology (e.g. by adding new properties to a concept, or by deleting concepts or instances).

Ontology frameworks parse ontologies to build the object models, or serialize ontologies into files or databases to persist them (57). A parser reads OWL and RDF(S) files which is for example represented in XML and builds the object model representation. A serializer generates the OWL and RDF(S) file based on the object model representation. Parsers and serializers therefore support the engineering cycle of ontologies (58). Through parsers and serializers interoperability between different frameworks is possible because they can exchange data using the serialized format like OWL (57).

If the ontologies are given in different ontology-representation languages and formats, parsers and serializers for each format can support these representations. Thus an ontology framework can also be used as conversion service between different ontology languages.

Additionally to serializer- and parser- persistence techniques, some ontology frameworks define adapters to different persistence layers. Examples are: flat-file storage, relational database storage or more sophisticated mechanisms as OWLIM (product name (59)), the OWLMemSchemaRepository SAIL (Storage and Inference Layer) for Sesame (60) – a framework for storing, inferencing and que-



rying RDF data (61). Through accessing proprietary persistence mechanisms semantic web programming frameworks can provide ontologies generated from data of different information systems (62). Hence legacy data can be integrated in ontology-driven applications (63).

Many frameworks provide more or less sophisticated functionality for developing ontology driven applications. Some frameworks are focused on the data or object model and support therefore the implementation of ontology driven applications that focus on application's logic. Others already support user-interface development for textual or graphical ontology editors (55).

Some ontology frameworks support only specific dialects of OWL, while others represent arbitrary ontology languages as abstract models. The advantage of abstract models is that they are independent of a specific syntactic ontology representation.

Ontology-driven applications have to react to changes in the ontology. Therefore it is useful if an ontology framework already supports event-listener or observer mechanisms which enable the implementation of interactive applications that react to ontology changes.

Please see section 2.4 for further details about ontologies.

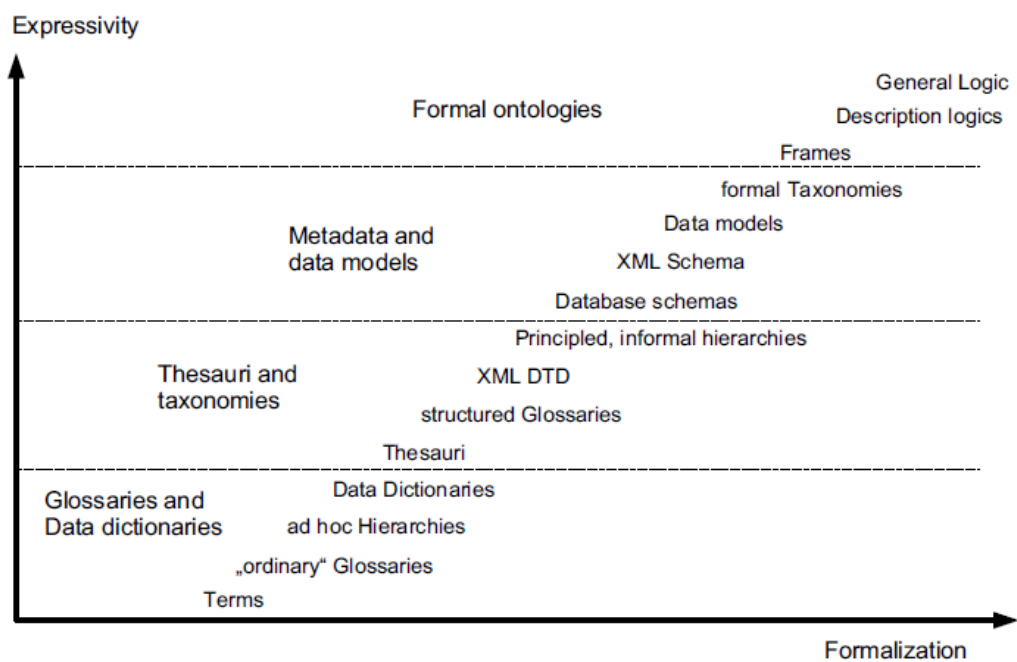
## **2.4 Ontologies**

Ontologies are often mentioned in the context of terminologies and archetypes. In fact they have a lot in common with terminologies and with archetypes; but ontologies cover much more. For example the idea of a reference model that is extended by an archetype model (dual model approach) came from ontology-influenced considerations for knowledge representation (28, 29, 36). Archetypes

and the dual model approach can be placed between *metadata and data model* and *formal ontologies* (Figure 8).

I already mentioned the terms *ontologies*, *terminologies*, *archetypes* and *dual model approach*. The dual model approach is described in section 2.1.2. Similar to the dual model approach also RDF(S) and OWL (see section 2.4.1 for a detailed description) ontologies focus on knowledge representation and knowledge sharing.

Figure 8 depicts a structuring of different types of ontologies according to (51).



**Figure 8: Types of ontologies (51)**

“The simplest ontologies are controlled vocabularies with a restricted list of terms, such as catalogs or identification systems, similar to dictionaries that are systematically arranged collections of words of a certain domain. Another form of specification are glossaries, i.e., a list of little-known terms with their meaning

description in natural language. Taxonomies are classifications of a particular subject field, i.e., a hierarchically organized controlled vocabulary. Thesauri offer additional semantic information regarding the relations between the terms, as they describe related classes as abstractions of objects and instances with identical properties, respectively. For doing so, basic term hierarchies use the idea of generalization and specialization by describing them as relations. Schemas define abstract record structures and thus organize them in accordance to conceptual models of the data to be represented. In formal ontologies the relation between super- and subclasses is strictly observed. Additional formalization is achieved, first, by including instances in the classification schemas and, second, by defining value constraints for the describing properties. The highest degree of expressivity is achieved by defining the properties by means of mathematical equations containing values of other properties or by logical statements in first-order logic, such as disjoint or inverse classes or part-of relations.”(51).

Several recent papers concerning EHRs emphasize formal ontologies for ontology driven applications (64-66). What does ontology driven mean? I would like to answer this question by comparing ontology driven with *database driven*. Databases and database schemas are well known and highly accepted representations of structured information. Database driven means, that the data structures (database schema) as well as data itself influence the application’s behavior. In other words the application design is derived from the data and from the data structure of the database. Ontology driven applications behave equally but it is the ontology that causes the application’s behavior or application design. More details about ontology driven applications are provided in section 3.2.2 and 4.3.

An ontology provides domain specific knowledge that is implementation and system independent. Therefore an ontology provides another layer of abstraction from an implementation point of view. Derived from ontology knowledge an ap-

plication may interact, may adapt its look and feel, or generally speaking adjust the application behavior to the domain knowledge provided through the ontology.

#### **2.4.1 Semantic web, resource description framework and web ontology language overview**

In the following I will describe the Resource Description Framework (RDF) and the Web Ontology Language (OWL). These languages are standards for ontologies and were defined by the World Wide Web Consortium (W3C).

The idea of the semantic web came up as the World Wide Web grew vast. Although search engines like Google, Yahoo, Bing, or others have implemented sophisticated statistical search algorithms, these search engines can only perform string searches and key word matches of page contents. What is missing can be described a *semantic search*.

Furthermore the decentralized organization of the World Wide Web caused different levels of heterogeneity of the represented information:

- different coding techniques (ASCII vs. Unicode, different file formats, etc.),
- different natural languages,
- and different structures of web pages.

These heterogeneities make it difficult to summarize semantically coherent but distributed information in the web. This problem can be described as *information integration problem*.

Further the user could be interested in information that is not explicitly given by the web, but can be inferred from other facts (which are possibly also distributed over the web). This problem could be described as *implicit knowledge problem*.

A solution for the mentioned problems is to structure the information machine-readable and -interpretable. This allows machines to interpret the information and represent it as coherent context. Basic needs for such a kind of solution are open standards. Open standards allow interchanging information between applications and systems. Furthermore open standards allow the specification of relations between distributed information. The ability of building relations between information as well as interchanging information can be described as *interoperability*.

To achieve the goals mentioned above, the W3C developed the basic standards RDF(S) and OWL (42). RDF(S) and OWL are ontology languages that are especially made to meet the idea of the semantic web. An ontology is an RDF(S) or OWL document that describes domain knowledge. Within an ontology information can be related to other information and through RDF(S)/OWL specifications domain knowledge can be interchanged with other systems. Furthermore, formal logical inference mechanisms facilitate to infer and extract knowledge from ontology specifications (67).

Rebstock describes in (51) further details of RDF(S) and OWL as follows:

A language often used for basic ontologies is the Resource Description Framework (RDF) (68). This is a specification of information – the resources on the Web. Resources can be described by attributes and their values, and also with the help of links to other resources (69). Metadata is described in triples, similar to elementary sentences in the form subject, verb and object (70). A triple contains <object, attribute, value>, where the attribute links the object (which is represents a resource) to the value. The latter are nodes, so that an RDF model forms a semantic network (71). An example for an RDF triple is shown in Figure 9.

The RDF code describes a certain web page represented by its URL. The description also comprises that the web page has a title and states it.

```
<rdf:Description rdf:about="http://www.w3.org/TR/rdf-syntax-grammar">  
  <dc:title>RDF/XML Syntax Specification (Revised)</title>  
</rdf:Description>
```

**Figure 9: Example for RDF notation**

Most widespread is the serialization of RDF in XML. The Resource Description Framework can serve as a fundamental general-purpose format for representing lightweight ontologies (71). With RDFSchema (RDF/S), the formal vocabulary for describing the semantics of the RDF elements used can be defined (72). RDFSchema includes the representation of classes and subclasses as well as properties for describing relations between information and instances, respectively (73). Its built-in main meta-classes are `rdfs:Class` and `rdfs:Property` (71). In this way, the concepts of an ontology can be represented. Therefore, RDF/S is suitable for classification hierarchies (74).

On this basis, the US Defense Advanced Research Projects Agency (DARPA) that also created the Internet's predecessor Arpanet, developed the DARPA Agent Markup Language (DAML) as a communication language for software agents. DAML was later combined with the Ontology Inference Layer (OIL) (69). OIL was intended to be an ontology language with formal semantics and extensive deduction possibilities (75). Together, the so-called DAML+OIL provided the basis for developing the Web Ontology Language (OWL). The Web Ontology Language is a W3C specification for creating ontologies with a formal representation language. In principle, it is a semantic markup language. Terms and their relations can be formally described by OWL in such a way that they become machine understandable. Technically, this language builds on the RDF syntax and also employs the triple model, but it features more powerful possibilities for building expressions

than RDF/S. As an extension of RDF and RDF/S, in OWL further language constructs are included, allowing for expressions formulated similarly to those with first-order logic. Vocabulary is added, providing the representation of relations between properties and classes, such as equality, cardinality and basic constraints (76).

The Web Ontology Language exists in three versions of different power which build on top of each other. Each lower and thereby less expressive dialect is a subset of the dialect above it. Accordingly, OWL Lite is the least expressive dialect and OWL Full the most substantial (77). These dialects facilitate the definition of constraints to different degrees, thus enabling deductions that are as comprehensive as possible (78). OWL Lite offers the functionalities of RDF/S with a few additions, but does not allow for metamodeling. Definitions and axioms can be represented as well as, to a small extent, properties for defining classes (75). This language serves mostly for creating taxonomies and lightly axiomized ontologies with basic constraints (73). OWL-DL, which is OWL with description logic, is the successor of DAML+OIL and adds expressive constructs from the field of description logics, such as transitivity of properties. With this dialect, maximum expressivity is possible while, at the same time, calculative correctness is ensured, which is of importance for automated reasoning (73). OWL Full, sometimes also called OWL Heavy, builds on top of OWL-DL without any restrictions and thus provides for metamodeling (79). This dialect also offers maximum expressivity, but it has additional potential for syntactical designing, similar to what is possible in RDF. However, due to these possibilities, calculative correctness is not fully ensured (73). For the same reason, this dialect can provide for handling classes as instances and vice versa. When coupling ontologies, this feature proves to be helpful (74).

## **2.5 OWL-S an ontology supporting process modeling and service interoperability**

In this section I will introduce OWL-S. “OWL-S is an OWL-based web service ontology, which supplies web service providers with a core set of constructs for describing the properties and capabilities of their web services in unambiguous, computer-interpretable form”(80). OWL-S is developed from the web service arm of the DAML program. DAML is described in section 2.4.1.

The aim of OWL-S is an ontology for web services with a set of standardized classes and properties that enable a service provider to create machine interpretable descriptions, and to enable a service requester to automatically find, invoke and composite the right services. Three exemplary and motivating scenarios are published on the OWL-S official website (81):

1. Automatic web service discovery
2. Automatic web service invocation
3. Automatic web service composition and interoperation

The following figures depict the structure of OWL-S, its classes and the important properties.

OWL-S comprises of 4 parts:

1. Service.owl (Service)
2. Profile.owl (ServiceProfile)
3. Process.owl (ServiceModel)
4. Grounding.owl (ServiceGrounding)

### **2.5.1 Service.owl:**



Figure 10 shows the top level ontology, Service.owl, of OWL-S and the relations to the other parts. The notations in this figure mean: black arrows are object properties and the labels of the arrows represent the names of the properties.

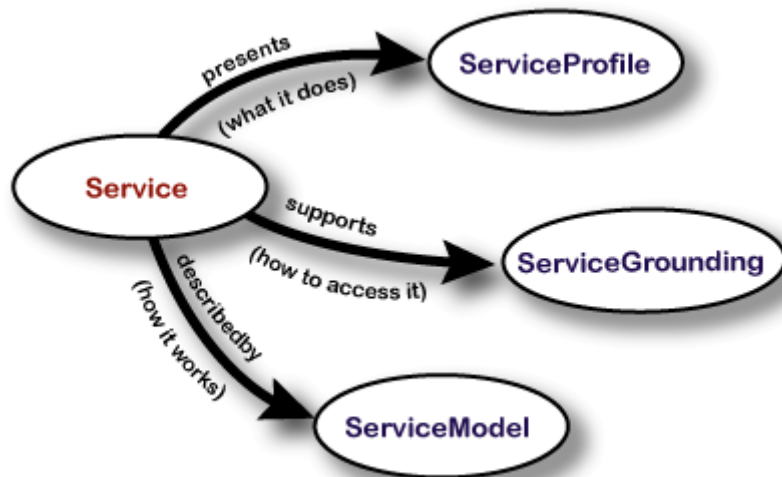


Figure 10: Top level of OWL-S ontology (81)

### 2.5.2 Profile.owl:

Figure 11 shows the Profile.owl ontology. The following descriptions are taken and summarized from the official OWL-S webpage (81). The class **ServiceProfile** provides a superclass of every type of high-level description of the service. **ServiceProfile** does not mandate any representation of services, but it mandates the basic information to link any instance of profile with an instance of service.

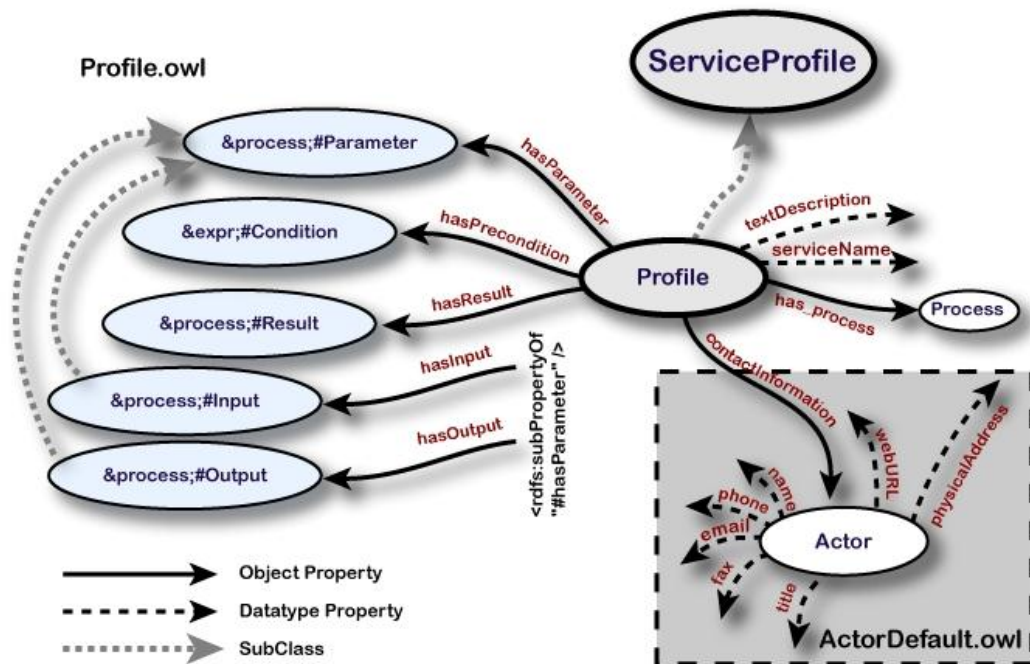


Figure 11: Selected classes and properties of the Profile (81)

Some properties of the profile provide human-readable information that is unlikely to be automatically processed. These properties include `serviceName`, `textDescription` and `contactInformation`. A profile may have at most one service name and text description, but as many items of contact information as the provider wants to offer.

The aim of `Profile.owl` is to provide the specification of what functionality the service provides and the specification of the conditions that must be satisfied for a successful result. `Profile.owl` represents two levels of functionality: first, the information transformation (represented by inputs and outputs) and second the state change produced by the execution of the service (represented by preconditions and results).

Here is a description of the functionality properties of `Profile.owl` as mentioned in (81):

- **hasParameter:** ranges over a parameter instance of the Process ontology. This means that the value the hasParameter property refers to, has to be an instance of Process or its subclasses (the meaning of range and domain is described in section 3.3.2). Note that the Parameter class models our intuition that inputs and outputs (which are kinds of parameters) are both involved in information transformation and therefore they are different from preconditions and effects. As a consequence, we do not expect this class to be instantiated. Its role is solely to make domain knowledge explicit.
- **hasInput:** ranges over instances of inputs as defined in the Process ontology.
- **hasOutput:** ranges over instances of type output, as defined in the Process ontology.
- **hasPrecondition:** specifies one of the preconditions of the service and ranges over a precondition instance defined according to the schema in the Process ontology.
- **hasResult:** specifies one of the results of the service, as defined by the result class in the Process ontology. It specifies under what conditions the outputs are generated. Furthermore, the result specifies what domain changes are produced during the execution of the service.

### **2.5.3 Process.owl:**

The reason why OWL-S supports process modeling is based on the assumption that a service can be seen as a process. The OWL-S notation is based on several works in different fields:

- work on standardization of planning languages (82)
- work on languages for business processes (83)

- work in programming languages and distributed systems (84, 85)
- work on process modeling and workflow technology National Institute of Standardization (NIST) Process Specification Language (PSL) (86)
- work on modeling verb semantics and event structure (87)
- work on modeling complex actions (88)
- and work on agents communication languages (89, 90)

A process cannot be seen as program being executed. A process is a specification of the interactions between a client and the service or a set of related services. There are two kinds of processes:

- Atomic process

An atomic process describes a service that has one input message (possibly complex) as request and returns one output message (possibly complex) as response. The purpose of an atomic process is to generate some new information based on the input information according to a certain state. Information generated is described by the inputs and outputs of the process.

- Composite process

A composite process can additionally maintain state information; each message a client sends proceeds a step through the process. The purpose of a composite process is additionally to the purpose of an atomic process to produce a change in the state. This transition is described by the pre-conditions and effects (results) of the process.

Figure 12 depicts the top level of the Process.owl ontology, its classes and relations. For more information about OWL-S and the Process.owl I will refer to the official OWL-S website (81).

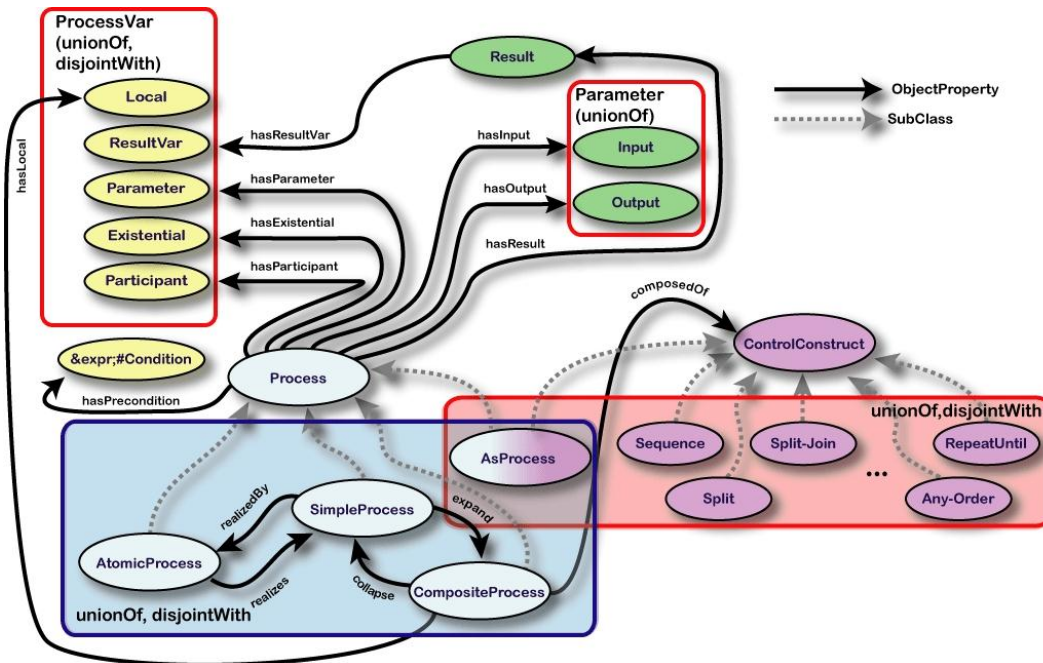


Figure 12: Top level of the process ontology (81)

### 2.5.4 Grounding.owl

The last part of OWL-S is the Grounding.owl. Through the grounding of a service we can specify the details of how to access the service. These details mainly address the protocol and message formats, serialization, transport, and addressing. A grounding can be seen as a mapping between abstract definitions provided by Profile.owl and Process.owl to a concrete specification of the service description elements that are necessary to interact with the service.

The basic idea of OWL-S is to implicitly describe the content of a message by describing the input and output properties of an atomic process. An atomic process describes the basic action, from which a larger process can be composed. Hence

an atomic process can also be seen as communication primitive of an (abstract) process.

The aim of the OWL-S grounding is to define how the (abstract) inputs and outputs of an atomic process can be realized as concrete messages defined in WSDL. Web Services Description Language (WSDL) is “an XML language for describing Web services. This specification defines the core language which can be used to describe Web services based on an abstract model of what the service offers. It also defines the conformance criteria for documents in this language” (91). Here it should be noted that the concept of OWL-S’ grounding is generally consistent with the WSDL’s concept of binding.

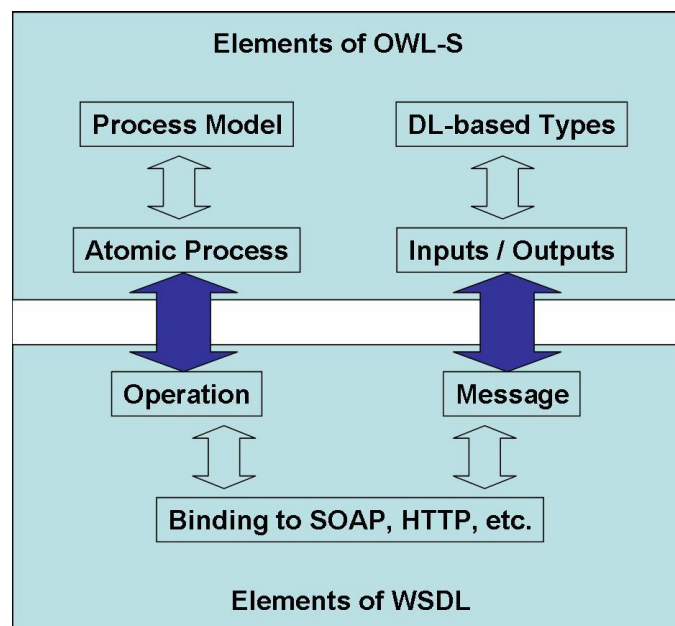


Figure 13: Mapping between OWL-S and WSDL (81)

Figure 13 shows the concept of the mapping between OWL-S and WSDL. Please have a look at the official webpage for more detailed descriptions and examples (81). In the result part of this work an implementation of OWL-S will be described.

## **2.6 The development environment**

In the following sections I will focus on the technologies to implement an ontology driven application.

First I will describe what the J2EE standard is about, and its reference implementation the Glassfish application server. Then I will drill down to the J2EE standard and explain how web applications can be implemented by programming Java ServerFaces, which is part of the J2EE standard. After explaining the Java ServerFaces specification I will give an overview of the web service capabilities of the Glassfish application server.

### **2.6.1 The Glassfish application server and the J2EE standard**

Glassfish is the name of an open source development project for the J2EE platform. J2EE stands for Java 2 Enterprise Edition and is a platform used for server programming. The J2EE specification is an umbrella specification that builds on the corresponding Java 2 Standard Edition (J2SE) specification (e.g. Java EE 5 requires Java SE 5) and includes several other API specifications (92). API stands for application programming interface. An API provides a particular set of instructions and rules that once implemented can be used by programmers to make use of the intended functionality (93). Some of these API specifications are: JDBC (Java Database Connectivity), email, web services, EJB (Enterprise Java Beans), servlets, JavaServer Pages, JavaServer Faces etc.

As these specifications are all API specifications an implementation provides a server environment to develop applications or implement access to (other) server content (e.g. databases, services). The latest implementation of J2EE provided by the Glassfish project is the Glassfish v3 application server. In the next sections (2.6.2 and 2.6.3) JavaServer Faces and Web Services are described in more detail.

The Glassfish v3 application server amongst others supports load balancing, is scalable and supports clustering. All these are important features for guaranteeing high performance, reliability and failsafe performance of modern IT systems.

To work with an application server, the software has to be installed on a computer or server machine. As Glassfish is a Java implementation it supports several operating systems like Sun Solaris, Windows, Linux and Mac OS which is Unix based. Additional installations are not necessary as the application server comprises all APIs of the J2EE 6 standard. This means, once the application server is installed application development and/or application deployment can take place.

### **2.6.2 Web applications and Java ServerFaces**

Java ServerFaces (JSF) is a framework-standard for developing web applications. JSF extends the Java Servlet and Java Server Pages technologies. JSF is part of the web technologies that come along with the J2EE standard. JSF facilitates the development of graphical user interfaces within web pages and dynamic web pages. Developing JSF content requires the Java Software Development Kit (SDK), a Servlet container implementation (e.g. Tomcat) or J2EE implementation like Glassfish and knowledge about the Hypertext Markup Language (HTML), the Hypertext Transfer Protocol (HTTP), Extensible Markup Language (XML) and the Java programming language (94).

### **2.6.3 Service oriented design and web services**

Basically there are two technologies to work with web services. One uses the Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL) and the Universal Description, Discovery and Integration (UDDI) to find and describe the web service and to exchange messages and data. SOAP, WSDL and UDDI are described in section 2.6.3.1.



The other technology uses the HTTP Unified Resource Locator (URL) as a representation of a web service and uses HTTP- POST, GET, PUT and DELETE to interact. This technology is called Representational State Transfer (REST). In this work I focus on SOAP/WSDL and UDDI based web service technologies.

### **2.6.3.1 Web services**

Web services are designed for web applications to interact with each other. Web applications are built around web browser standards like HTML. Web services enable the interoperability between these web applications. With web services an application can publish its function to the rest of the world (95).

Web services are based on XML and HTTP. XML is a language for encoding electronic documents and can be used between different platforms and programming languages (96). HTTP means hypertext transfer protocol and is widely used among the internet.

Web services are application components that can be communicated using open protocols. They are self-contained and self-describing and can be discovered using UDDI. UDDI will be explained later in this section. Web services can be called and used by other applications (95).

Web services consist of three elements:

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

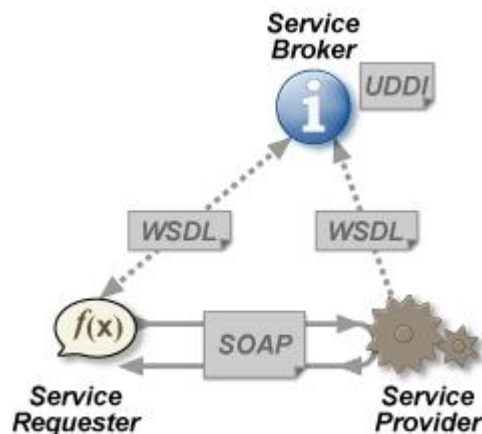


Figure 14: Web service architecture(97)

Figure 14 shows the web service discovery by UDDI, the web service description provided by WSDL and the web service call through SOAP. SOAP, UDDI and WSDL are described next:

### 2.6.3.2 Universal description, discovery and integration

UDDI means Universal Description Discovery and Integration and is a directory service where companies can register and search for web services (95). UDDI communicates via SOAP and is a directory of web service interfaces described by WSDL.

### 2.6.3.3 Web services description language

WSDL means Web Service Description Language and is a language to describe web services via an XML document. A WSDL document consists of the following four elements (98):

**<types>**

This element describes the data types that can be used by a web service. XML Schema syntax is used to define data types (98).

#### **<message>**

Each message can consist of one or more parts and defines the data elements of an operation. The parts of a message can be compared to the parameters of a method call in a traditional programming language (98).

#### **<portType>**

The portType describes the web service, the messages that can be called and the operation that can be executed. The portType element is the most important element and can be compared to the class in a traditional programming language (98).

#### **<binding>**

The binding element defines the protocol details and the message format for each port.

#### **2.6.3.4 Simple object access protocol**

SOAP means Simple Object Access Protocol and was designed to communicate between applications using the HTTP protocol. Each application can use its own technology and programming language and SOAP to communicate with each other. SOAP is a format for sending messages, it is platform and language independent and XML based (99). Roughly formulated WSDL describes the web service and its function, and SOAP communicates the data and instances between the web service requester and the web service provider as shown in Figure 14.

## 3 Methods

A practical advice to keep things simple in computer science says: “divide and conquer” and it means, that if you have a big complex problem you can solve this problem by splitting the problem in many less complex problems, solve them separately and finally rearrange the solutions until the whole problem is solved. This method is called decomposition.

For my work to solve the problem of building an ontology driven EHR for a distributed environment, it is necessary to split the work and solve the problems separately. I will capsule complex contents and address them separately.

### 3.1 The GLIF methodology for defining electronic clinical guidelines

In this section I will to describe the steps and methodology necessary for defining an electronic guideline using GLIF3. In section 4.4.1 I will refer to the current section to show that my methodology of defining health delivery and integrated care processes follows the same requirements as GLIF3. I will not provide a complete description of the GLIF3 object model. For more details about the object model please refer to the GLIF3 specification (24). I will provide an overview about the methodology and the entities that are used when creating a guideline using GLIF3.

Basically, creating a guideline in GLIF3 is similar to specifying an algorithm (the corresponding class of the object model is called Algorithm). The guideline’s algorithm can be seen as the flowchart of guideline steps. The guideline steps cover the knowledge of the guideline. A guideline step can be one of the 5 following steps:

- an action step,

- a decision step,
- a branch step,
- a synchronization step or
- a patient state step.

The attributes:

- next\_step (in case of an action step, synchronization step or patient step)
- branches (in case of a branch step)
- options (in case of a decision step; with the attribute destination)

indicate which step(s) is/are next in the flow chart.

Let us have a close look on the different steps to clarify which attributes lead to one's next step.

### **3.1.1 Action step**

Basically an action step models a clinical action that has to be performed when it becomes active within the flowchart. An action step can have numerous facets:

1. An action step can be “medically oriented action” that refers to a medical term and represents a typical guideline recommendation. These actions describe clinically relevant actions.
2. An action step can be a “get data object action”. The purpose of this action is to obtain a value of a data item from a user entry or an EHR and store it into a variable. This kind of action is a programming oriented action specification and describes guideline-flow-relevant actions.
3. An action step can be a “subguideline action” that contains the details of a high-level action. A high-level action can be a (sub) guideline this action

stands for. It is also a programming oriented action specification and they also describe guideline-flow-relevant actions.

There are some other programming oriented action specifications necessary for data communication and data assignment with and from an EHR system. Action steps are necessary for formulating the guidelines semantics.

Each action class inherits the next\_step attribute. The next\_step\_attribute refers to the type of the general class of Guideline\_Step. All (guideline) steps are derived from this class.

### **3.1.2 Decision step**

The decision steps define the control flow of an electronic guideline relating to the condition within the decision step. GLIF3 provides a sophisticated model of different decision steps for specifying deterministic and non-deterministic decisions.

A decision step is linked to various decision options (attribute option refers to the Decision\_Option class). Each of the options indicates a possible decision and is associated with a destination (destination attribute in Decision\_Option class). The destination attribute refers to a Guideline\_Step class, to point to the next step in the guideline if the condition evaluates to true.

In the following I will summarize the different kinds of decisions which can be modeled with GLIF3:

1. Deterministic decisions are mutually exclusive. If the strict\_rule\_in attribute evaluates to true, then the control flows to the guideline step that is specified by the decision option's destination.
2. There are three kinds of non-deterministic choices:

- a. The rule-in-choices model criteria as choices the user can pick. These criteria help the user choosing one of the decision options. The choices do not have to be mutual exclusive. The criterion that fits best causes the decision. Some of these choices however can be strict rules. As an example if the decision evaluates to the rule a patient has “allergy to penicillin” than penicillin must not be prescribed. A strict rule has a higher priority than a choice rule.
- b. Weighted choices are different criteria, each associated with a certain ranking or weight. The sum of all weights for each criterion has to be 1. The higher the value of the weight of the choice, the higher its rank. The weight of a criterion helps the user to determine the right decision during runtime.
- c. The last of the non-deterministic choices is a utility choice. It represents a node in an influence diagram or decision analysis tree.

### **3.1.3 Branch step**

At a branch step the path of the flowchart splits up into several branches. A branch step allows to model concurrent guideline steps. The steps directly following the same branch step must occur in parallel.

### **3.1.4 Synchronization step**

Synchronization steps are basically the opposite of branch step. A synchronization step joins different paths of the control flow together. The synchronization step specifies which of the paths - that are joining up at a synchronization step - have to be completed before the control can move to the next step.

### **3.1.5 Patient state step**

A patient state step is used for two purposes. First a patient state step serves as guideline entry point. For example a patient comes back to the clinic at patient state A. The second purpose of the patient state point is to describe the patient state that has been achieved by processing the previous guideline steps. This facilitates to match a patient state within a guideline. Therefore a guideline does not have to start at the beginning of the guideline itself, but can start anywhere in the middle at the corresponding patient state step.

### **3.2 Building semantic web applications**

In (100) Holger Knublauch describes some principals about ontology driven software development. He describes a semantic web example scenario from the tourism domain: Providers of travel-related services such as holiday activities and accommodations advertise their services on the semantic web in a way that intelligent agents can find them dynamically. The agents could subsequently make suggestions on vacation planning.

As an example Figure 15 shows the travel ontology which could be defined by a standards body of the tourism industry, the geography ontology could be provided by a government agency. Both ontologies would be published on fixed URI as OWL files. Based on the expressiveness of OWL, it is possible that local providers publish extensions to an ontology with their logical characteristic. In Figure 15 the HeliBungeeJumping class extends the BungeeJumping class.

A base ontology like the travel ontology would allow providers to publish metadata about their services and contact information. Providers would instantiate the classes from the ontology and publish the resulting individuals as OWL files on their web sites. Then, a Semantic Web service specialized in vacation planning could send out a crawler agent to collect the available activities. If a



user then asks for an exciting adventure destination, the agent could exploit the categorization of the ontology hierarchy to find suitable matches, and call auxiliary Web Services via the links into the geography ontology (100).

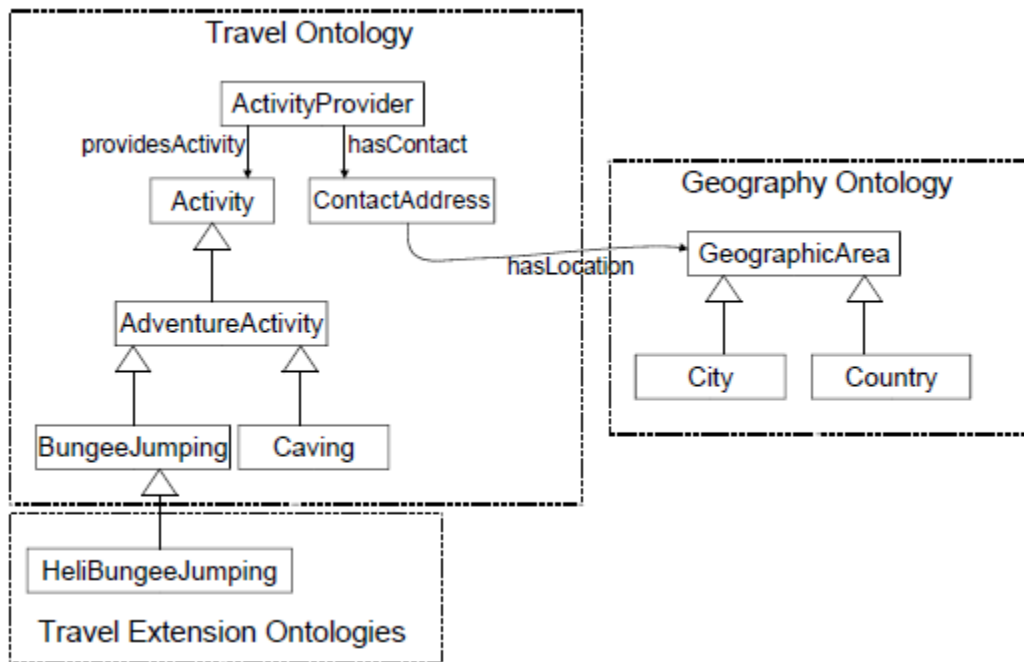


Figure 15: Ontologies from an example Semantic Web scenario (100)

### 3.2.1 Architecture of a semantic web application

The major part of the application logic is implemented in a conventional object-oriented language like Java. Some systems might manage data bases, sessions and user interfaces. The application needs to access the ontology individuals as objects. These objects are exchanged between the application and other services or the user interface for representational issues. If an ontology is known to the application in advance, custom tailored Java classes can be built that access a dynamic object model like Jena to read and write OWL code. Custom tailored classes allow programmers to attach methods to these classes, leading to a cleaner object oriented design pattern. Ontologies are also used to represent the

background knowledge needed by the application to compute its tasks. This knowledge and its structure are defined by base classes of some core ontologies. As one idea of the Semantic Web is to extend existing resources, the base classes can be instantiated or subclasses arbitrarily by external ontology providers. The external ontology extensions can only be used by generic reasoning engines. The base classes can be accessed by specific application code of the executable system to interpret the ontology (100).

Figure 16 shows a software architecture for an application that finds appropriate holiday destinations for a customer.

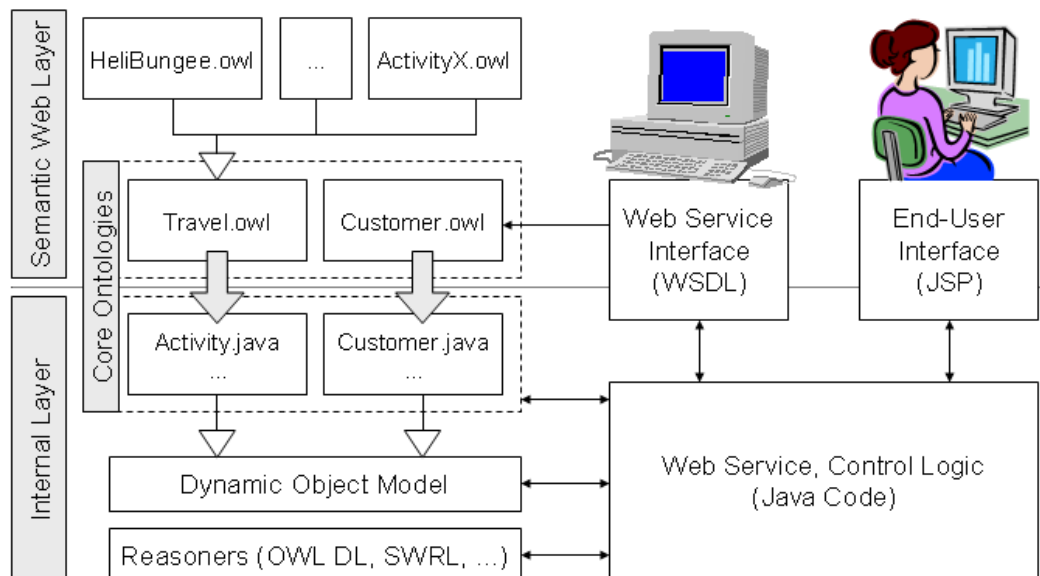


Figure 16: Example Architecture (100)

### 3.2.2 Ontology-driven software development

As already illustrated in Figure 16 a semantic web application consists of two separate but linked layers. A Semantic Web layer makes ontologies and interfaces available to the public. The Internal Layer consists of the control and reasoning mechanisms. The artifacts of the semantic web layer have to meet higher quality

standards than those of the Internal Layer. This is because the artifacts of the Semantic Web layer are shared with other applications. For triggering the internal behavior e.g. the outcome of a reasoning algorithm, the models of the Semantic Web layer are interpreted. A lot of internal code fragments can be generated from higher-level models, consisting of generic libraries for reading and accessing the ontology data (100).

### **3.3 Ontology design and best practice**

Many methodologies of software engineering can be reused for ontology engineering. In the next sections I will give an overview about the methodologies necessary for this work.

#### **3.3.1 General design principals**

Devedizic described in (101) that methodologies for ontology engineering can be borrowed from the software engineering disciplines shown in Figure 17.

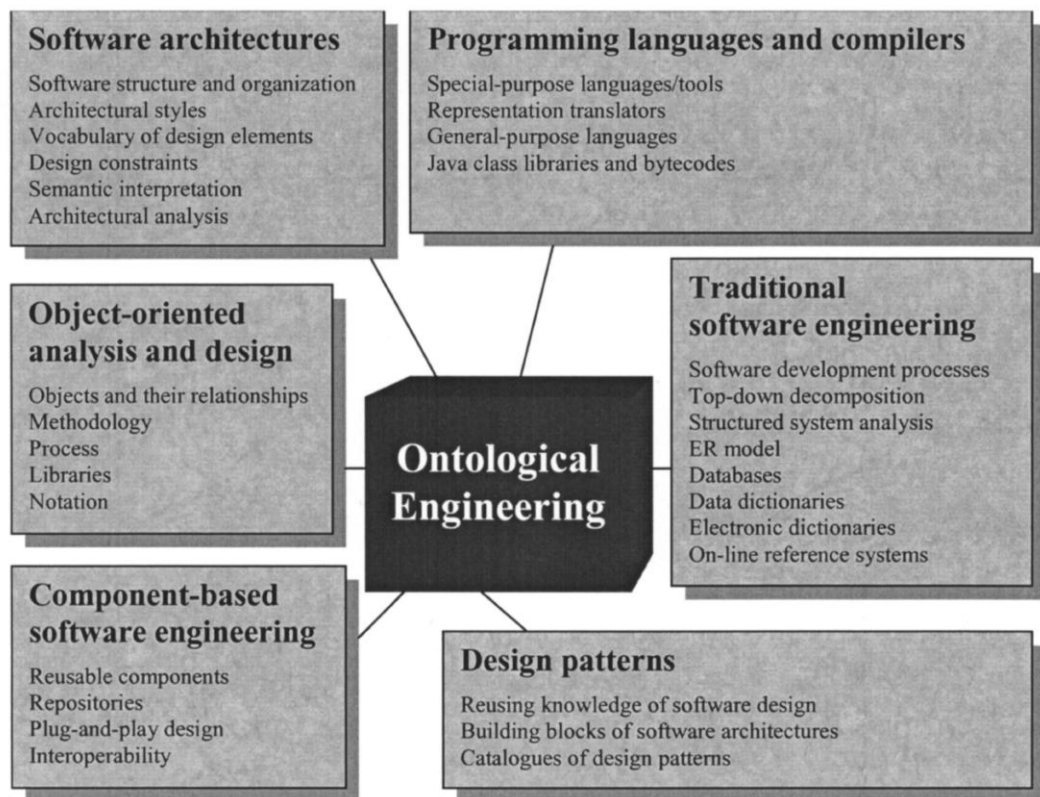


Figure 17. Software engineering disciplines useful to ontological engineering (101)

Figure 17 illustrates several areas, design and engineering methods can be borrowed from.

In this work I focus on the areas of object oriented analysis and design as well as on design patterns for engineering ontologies. Gruber claimed in (28) some common principals for ontology engineering and Gruber mentions that have to be designed. Therefore ontology designers have to make ontology decisions. To make the right decisions Gruber suggests the following design criteria for ontologies:

- **Clarity**: Defining concepts often arise from social situations or computational requirements. The definition of a concept should be independent of the social or computational context. The intended meaning of the on-

ontology terms should be effectively communicated by the ontology. Logical axioms should be stated for the definition whenever possible. The definitions should be documented with narrative natural language.

- **Coherence:** Consistent definitions should allow inference if they are logically coherent. It is necessary that examples defined in natural language examples do not contradict logical predicates inferred from the ontology axioms. If this is not the case an ontology is incoherent.
- **Extendibility:** An ontology should be designed to be extended by other ontologies. Other ontologies designers should be able to specialize and extend the existing ontology vocabulary without revision the existing ontology definitions. Therefore the ontology should provide a design that allows to extend and specialise the ontology.
- **Minimal encoding bias:** An extended encoding bias occurs when ontology definitions are made purely for the convenience of notation or implementation. The ontology design should be oriented on the information and knowledge representation without depending on particular implementations that use the ontology. Encoding bias should be minimized to allow different styles of representation and to facilitate implementations in different knowledge sharing systems.
- **Minimal ontological commitment:** To model the scenario of interest an ontology should make as few claims as possible. This allows freedom to all parties specializing, extending or instantiating the ontologies to their needs. The consistent use of axioms and vocabulary is necessary for ontological commitment. Ontological commitment can be minimized by defining the weakest theory (allowing the most models) and defining only the essential terms to communicate the knowledge of the ontology.

### 3.3.2 How to design an ontology

These design principals are very general in describing the major points to which a designer has to pay attention. The following aspects suggest more specific claims for web ontology design. Noy and McGuinness proposed in their Protege tutorial several criteria for web ontology development (102):

First Noy and MacGuinness mentioned 3 principles for ontology development:

1. There is no one correct way to model a domain— there are always viable alternatives. The best solution almost always depends on the application that you have in mind and the extensions that you anticipate.
2. Ontology development is necessarily an iterative process.
3. Concepts in the ontology should be close to objects (physical or logical) and relationships in your domain of interest. These are most likely to be nouns (objects) or verbs (relationships) in sentences that describe your domain.

Then Noy and MacGuinness described in 7 steps the procedure for defining an ontology:

1. Determine the domain and scope of the ontology

When we are starting to design an ontology we should first define the domain and scope of the ontology. The answers of the following basic questions could be helpful:

- What is the domain that the ontology will cover?
- For what we are going to use the ontology?
- For what types of questions the information in the ontology should provide answers?

- Who will use and maintain the ontology?

Even if the answers to these questions change during the ontology-development process these questions help limit the scope of the model.

In this work I focus on two ontologies. The focus of the first ontology is to represent the ISO 13606 standard and to provide the structure of an EHR\_Extract (see section 4.5.4.2). The focus of the second ontology is to provide a process model that describes a communication procedure between systems (see section 4.4.1).

## 2. Consider reusing existing ontologies

It is almost always important to consider reusing existing ontologies and extending for particular domain and task. If our system has to interact with other systems which are already committed to an ontology, it is almost a requirement to reuse this ontology.

There are libraries for ontologies on the Web and in literature. E.g.: the Ontolingua ontology library (<http://www.ksl.stanford.edu/software/ontolingua/>) or the DAML ontology library (<http://www.daml.org/ontologies/>). There are also some publicly available commercial ontologies (e.g. UNSPSC ([www.unspsc.org](http://www.unspsc.org)), RosettaNet ([www.rosettanet.org](http://www.rosettanet.org)), DMOZ ([www.dmoz.org](http://www.dmoz.org))) so you have to consider license clauses, too.

An ontology I reused for process descriptions is called OWL-S. I am going to describe OWL-S a later in section 2.5. I also reused the ontology described in section 5.3 to build an ontology repository based on ISO 13606.

## 3. Enumerate important terms in the ontology

When it comes to design an ontology it is useful to write down all terms we would either like to make a statement or to explain it to the user.

- What are the terms we would like to talk about?
- What properties do those terms have?
- What would we like to say about those terms?

It is important to get an comprehensive list of terms initially, without worrying about overlap between concepts these terms represent, relations among the terms, or any properties that the concepts may have, or whether the concepts are classes or slots.

Before I extended the OWL-S and the archetype ontology mentioned in section 5.3 I had to consider the focus the extended ontologies should meet. The OWL-S ontology was extended with the GLIF3 knowledge model described in section 4.4.1. Therefore I had to enumerate the terms which were missing in the OWL-S ontology. For the archetype ontology I also enumerated the missing terms to meet the criteria of ISO 13606 see section 4.5.4.2 for more details.

The following two steps are closely intertwined. It is hardly possible to do one of them first and then do the other. These two steps are the most important steps in the ontology development-process. Typically we create a few definitions of the concepts in the hierarchy and then continue by describing properties of these concepts and so on.

#### 4. Define classes and class hierarchy

There are several possible development process approaches for defining class hierarchies (103):



- Top-down: Here we start defining the most general concepts in the domain and continue with the specialisation of the concepts afterwards.
- Bottom up: here we start defining the most specific classes, the leaves of the hierarchy and continue with grouping these classes into more general concepts.
- A combination of bottom up and top down. We define the more salient concepts first and then generalize and specialize them appropriately.

It depends on the domain and the developer which approach is preferred. Each approach has its advantages and disadvantages for example if a developer has a systematic top down view of the domain it could be easier to choose the top-down development process. Basically the combination approach is the most appropriate one, since the concepts “in the middle” tend to be the more descriptive concepts in the domain (104).

Based on the enumeration of the previous step I added the classes to the OWL-S ontology and the archetype ontology to extend them. After creating the classes I also added properties and its facets to the ontologies (see section 4.4.1 and see section 4.5.4.2 for more details). Properties and facets are described in the next two steps.

#### 5. Define the properties of classes – slots

After defining the classes we have to define the internal structure of concepts. For each property we have to determine which class it describes. These properties become slots attached to classes. Generally there are several types of object properties:

- “intrinsic” properties describe the nature of the object

- “extrinsic” properties provide extended descriptions of the referenced object
- Parts, if the object is structured (physical and abstract parts (e.g. course of a meal))
- Relationships to other individuals

All subclasses of a class inherit the slots of their superclass. We should attach a slot to the most general class that can have that property.

#### 6. Define the facets of the slots

Facets of the slots define the value types, allowed values, the number of the values (cardinality), and other features of the values the slot can take.

- Slot cardinality
  - single cardinality (allowing at most one value)
  - multiple cardinality (allowing any number of values)
  - minimum and maximum cardinality, describing the number of slot values more precisely. Also maximum cardinality of 0 is possible to describe that a slot must not have any values of a certain instance.
- Slot-value types
  - String: describing slots with string values
  - Number: describing slots with numeric values
  - Boolean: simple true-false flags
  - Enumerated: list of specific allowed values
  - Instance type slots: define relationships between individuals.
- Range of a slot
  - The range specifies the allowed classes (for an instance type slot) to which the slot references via its properties. It is also possible to

restrict the range of a slot when the value type of a slot is attached to a particular class.

- Domain of a slot
  - The domain of a slot defines the class to which a slot belongs. An example for domain and range: Person livesAt Address. The class Person is a domain, and the class Address is the range of the property livesAt. There are some rules defining the domain and the range of a slot: When defining a domain or a range for a slot, find the most general classes or class that can be considered as the domain or the range for the slots. On the other hand, do not define a domain and range that is overly general. All the classes in the domain of a slot should be described by the slot and instances of all the classes in the range of a slot should be potential fillers for the slot. Do not choose an overly general class for range (i.e., one would not want to make the range THING) but one would want to choose a class that will cover all fillers. If a list of classes defining a range or a domain of a slot includes a class and its subclass, remove the subclass. If a list of classes defining a range or a domain of a slot contains all subclasses of a class A, but not the class A itself, the range should contain only the class A and not the subclasses. If a list of classes defining a range or a domain of a slot contains all but a few subclasses of a class A, consider if the class A would make a more appropriate range definition.

## 7. Create instances

The last step is defining instances of classes. When creating individual instances we have to consider:

- First, choosing the right class

- Subsequent creating the individual instance of that class
- And filling the slot values.

Instances of the archetype ontology are created using a tool described in section 5.3 that maps ADL to OWL considering semantic relations. In Instances within the EHR-ontology (which are instances of the EHR\_EXTRACT class) are created at runtime of the system. And instances of the extended OWL-S ontology are created using Protégé (for more details to Protégé see section 3.4.1).

## **3.4 Tools and application programming interfaces**

### **3.4.1 Protégé – an ontology engineering toolset**

Protégé was developed at the Stanford University (CA, USA) and is an application for designing and developing ontologies. Protégé comprises a graphical editor and several other plug-ins for manipulating ontologies. The most relevant plug-in for developing OWL ontologies is the Protégé OWL plug-in (105). It comprises the Protégé OWL API, an ontology framework based on Java. Protégé is available under the Mozilla Public License (106). More on the Protégé OWL API in the subsequent section 3.4.1.3

As Protégé is a tool for developing ontologies it provides two different editors for ontology development.

#### **3.4.1.1 Protégé Frames**

Protégé-Frames implements a knowledge model which is compatible with the Open Knowledge Base Connectivity protocol (OKBC)(107). In this model, an ontology consists of a set of classes organized in a subsumption hierarchy to represent a domain's salient concepts, a set of slots associated to classes to describe

their properties and relationships, and a set of instances of those classes - individual exemplars of the concepts that hold specific values for their properties (108).

#### **3.4.1.2 Protégé OWL editor**

The Protégé-OWL editor is an extension of Protégé that supports the Web Ontology Language (OWL). OWL is the most recent development in standard ontology languages, endorsed by the World Wide Web Consortium (W3C) to promote the *Semantic Web* vision”(109). "An OWL ontology may include descriptions of classes, properties and their instances. Given such an ontology, the OWL formal semantics specifies how to derive its logical consequences, i.e. facts not literally present in the ontology, but entailed by the semantics. These entailments may be based on a single document or multiple distributed documents that have been combined using defined OWL mechanisms (110).

#### **3.4.1.3 Protégé OWL API**

The Protégé OWL API (application programming interface) is based on Jena. Jena is an RDF and OWL parsing library. The Protégé OWL API provides the functionalities to access RDF/OWL code and mapping the individuals to Java objects. It also facilitates to serialize Java objects to RDF/OWL code. Furthermore different reasoner machines can be instantiated by the API. A reasoner allows to automatically interpret the OWL code and represents the knowledge derived from the interpretation. Basically the Protégé OWL API consists of ready to use Java libraries that facilitate the handling of OWL files and databases.

#### **3.4.2 Link EHR-ED archetype editor**

Link EHR-ED is an open source archetype editor (111, 112). The following descriptions are taken from the LinkEHR webpage (112) and slightly adapted to meet the focus of this work.

The LinkEHR-ED archetype editor was developed to design, create and evaluate clinical information structures. These information structures often already exist at health organizations. The output of LinkEHR-ED is based on standards and formal representations which describe the clinical meaning. These standards and representations are:

- ISO 13606 and OpenEHR which both use the dual model approach for their EHR architecture.
- LinkEHR-ED creates files written in ADL as formal representation of an archetype written in ADL. More details about archetypes, ADL and the dual model approach can be found in section 2.1.

For more information about LinkEHR please refer to (112).

### **3.4.3 OWL-S application programming interface**

The following description is taken from the homepage (113) of the OWL-S API which comes along with the MIT licence which is basically an abbreviation of an open source licence (114):

The OWL-S API is a library written and intended for the Java programming language. The purpose of the OWL-S API is to programmatically access OWL-S ontologies. The API comprises functionalities like a process execution engine to interpret OWL-S processes. It also facilitates to call the Web Service(s) that are described by an OWL-S process. The API can execute the control constructs of an OWL-S process which can be assembled of the following building blocks: Choice,

Sequence, AnyOrder, Split, Split-Join, IfThenElse, RepeatUntil, and RepeatWhile. The execution of a process can be monitored by means of ProcessExecutionMonitor.

To formulate conditions the OWL-S API supports the **Semantic Web Rules Language (SWRL)** and **SPARQL Protocol and RDF Query Language (SPARQL)**. The library also comprises an OWL reasoning engine called Pellet. Because the OWL-S API is based on JENA also other reasoning engines can be integrated. JENA is an open source project written in Java that provides API functionalities to read and write RDF and OWL files.

The structure of the Java classes and the names of the accessor methods are designed to match the names of the classes and properties of the OWL-S ontology classes and properties.

#### **3.4.4 The NetBeans integrated development environment**

NetBeans is an integrated development environment (IDE) for developing several programming languages like Java, JavaScript, PHP, Python, Ruby, Groovy, C, C++ and others.

NetBeans itself is a Java implementation and runs everywhere a Java virtual machine (Java runtime environment) is installed. The Java development kit (JDK) is required for Java development functionality, but is not required for developing in other languages.

For developing J2EE applications including Web Services and JSF-web applications a J2EE compliant application server has to be installed. The integration between several J2EE compliant application servers and the NetBeans IDE is very convenient. The integration comprises code generation, server-side debugging, control-

ling server functionality through NetBeans interfaces and a lot more. Especially the Glassfish J2EE application server integration is very well supported. For example: NetBeans provides a preconfigured all in one installation for J2EE development based on the Glassfish J2EE server.



## 4 Results

I start this chapter with an illustration of the idea behind this work in section 4.1. Subsequently in section 4.2 I will describe the wound management scenario. In section 4.3 I present the results of how to work with ontologies programmatically. In section 4.4 I introduce the wound management process and the wound management archetypes and finally in section 4.5 I will provide insights of the system architecture.

### 4.1 The idea

Figure 18 illustrates the idea behind this work. Figure 18 consists of three columns:

- A service column that illustrates the EHR\_EXTRACTs which are accessed by services.
- A runtime column that illustrates the communication process which is interpreted during the runtime of a calling system. The calling system dynamically composites the service calls dependent on the interpretation of the process description.
- And a result column that illustrates the goals of the scenario which are that each calling system is able to trace the diagnosis and treatment process, may implement reminders and warnings and may implement a dynamic system answer from interpreting the communication process state.

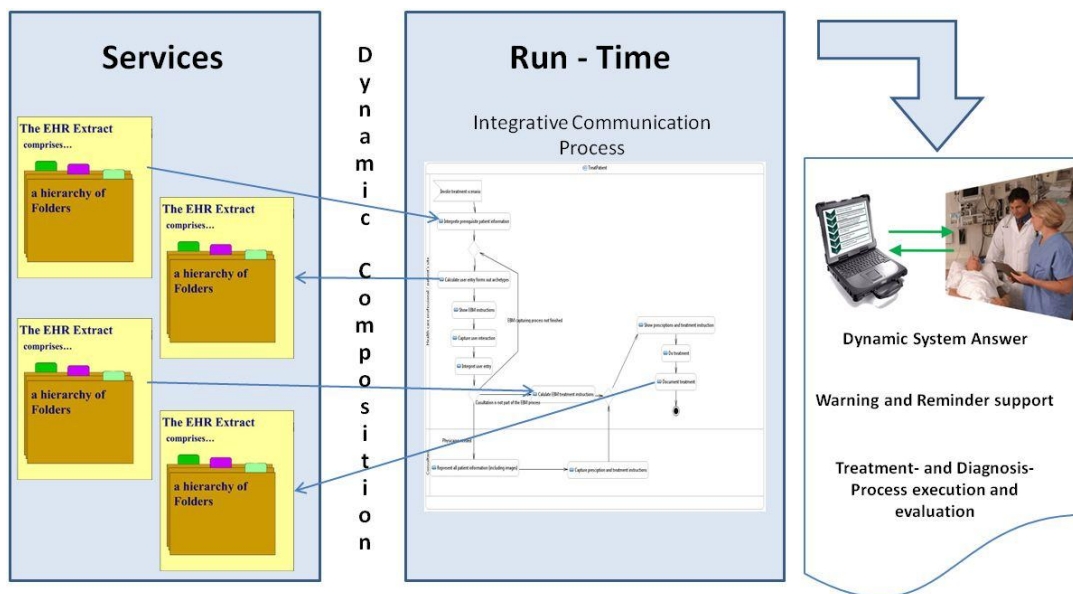
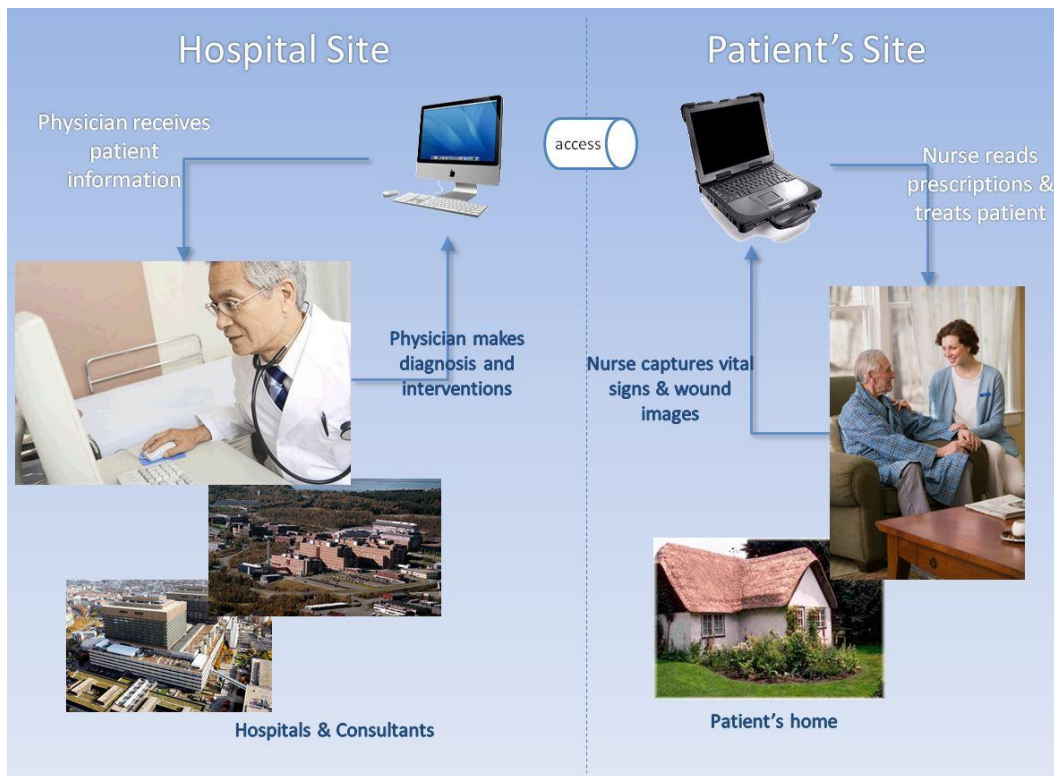


Figure 18: Integrative communication process embedding EHR-services

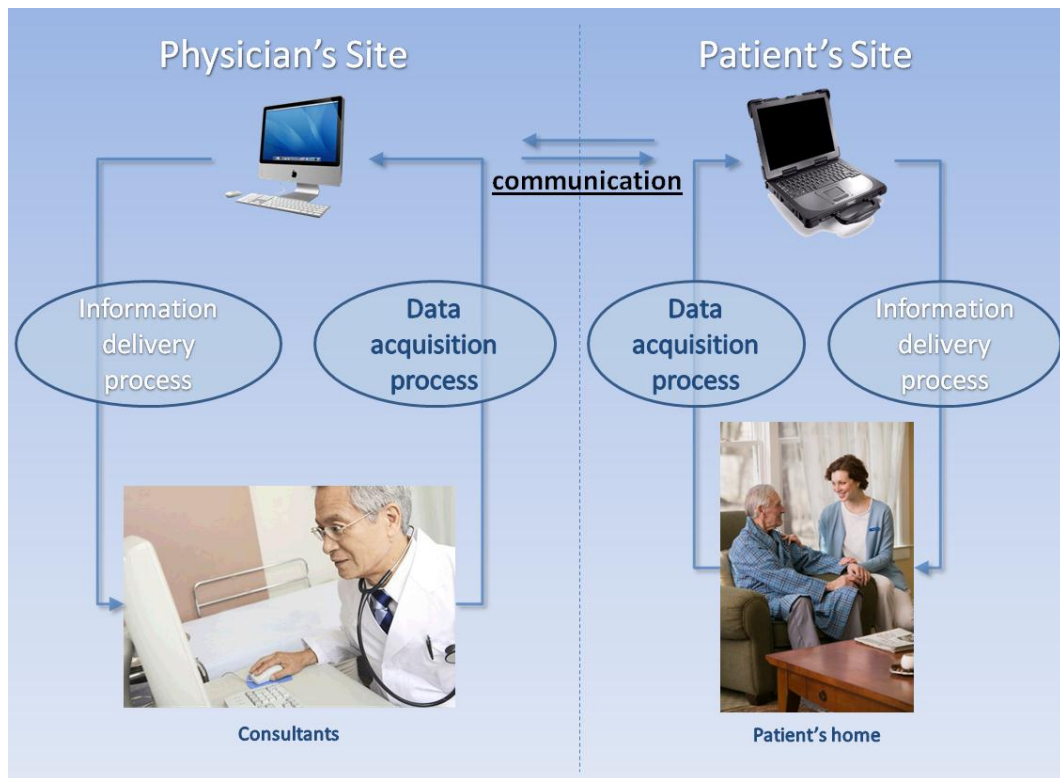
## 4.2 The wound management scenario

Figure 19 depicts the telemedical wound management scenario at the most concrete level. The aim is to find the building blocks of a telemedicine scenario. Subsequently I will to abstract these building blocks to condense the IT relevant aspects of these building blocks. On the right side in Figure 19 there is the patient, and a nurse that visits the patient to capture his vital signs and wound images. The nurse also treats the patient's chronic wounds. On the left side there is the specialist who receives and reads the vital signs and wound images. He prescribes the interventions. The whole scenario is designed to meet the legal requirements which are that the responsibility of the physician is to document the prescription of the interventions and the responsibility of the nurse is to perform the treatment and initially document the patient's health status.



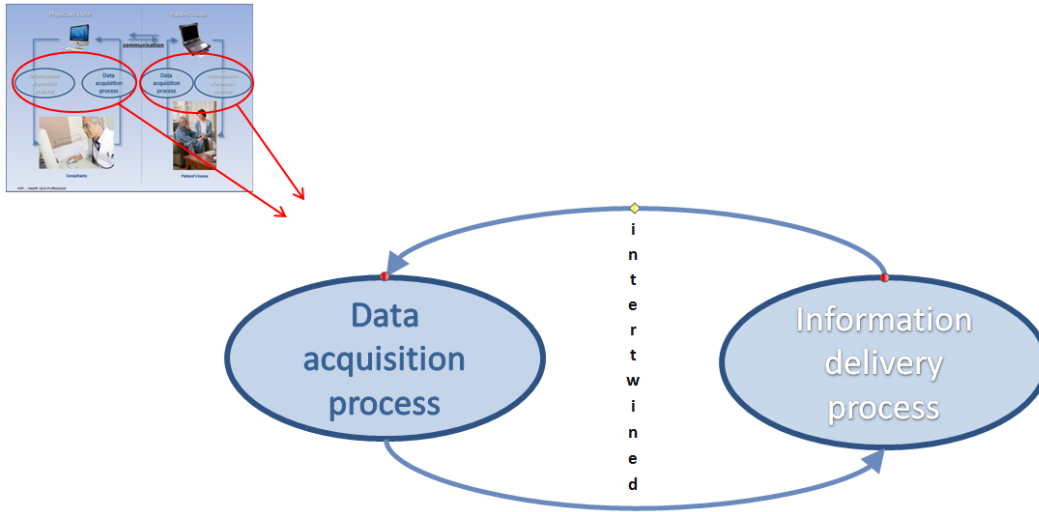
**Figure 19: a telemedicine support for outpatient wound management**

Figure 20 illustrates the abstract building blocks of the scenario in telemedical wound management scenario. These major building blocks are *data acquisition process*, *information delivery process* and *communication*. There are still some components missing that define the telemedicine scenario for treating patients with chronic wounds like the communication process. Let us have a closer look on the abstract building blocks in Figure 20.



**Figure 20: abstractions of the telemedical wound management scenario**

Figure 21 shows these abstractions in more detail. The data acquisition process and the information delivery process depicted in Figure 21 are strongly intertwined with each other. This means a closed cause and effect relation between these two processes. For example: if the nurse captures the patient information, the system validates the information and starts the information delivery process either back to the nurse representing reminders or input mistakes or to the physician who receives the patient information.



**Figure 21: Data acquisition and information delivery process**

To implement a scenario as depicted in Figure 19, two types of knowledge have to be added to the abstractions shown in Figure 20:

- knowledge about the required patient information (archetype based information) and,
- knowledge about the communication process (diagnosis/treatment) that defines a concrete scenario.

Both types of knowledge are necessary to sufficiently and completely describe the scenario of Figure 19.

However before we drill down to the application of the wound management scenario we have to do some basic considerations that help us to understand the generic concept of this work.

### **4.3 The basics of an ontology driven application**

An ontology driven application is an application which derives its functionality, behavior, look and feel, or data-definitions through interpreting ontology knowledge. However, before we start designing an ontology driven application some considerations have to be done.

This section (section 4.3) is the conceptual part of this work. First I present a schematic representation of the components of an ontology driven application model (ODAM) shown in Figure 22. Then I will describe the relations between the ODAM and the 13606 EHR communication standard. And finally in this section I describe how processes can be applied for ODAM.

#### **4.3.1 The ontology driven application model - ODAM**

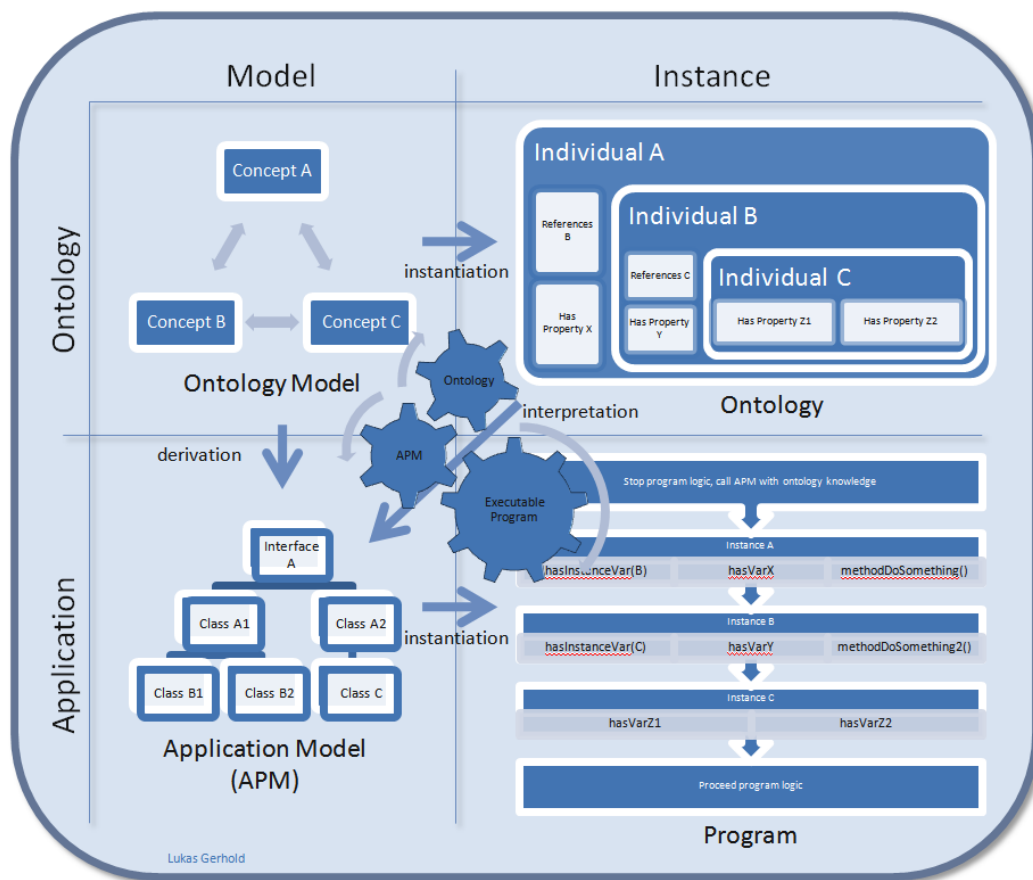
In this section I deal with the structure of an ontology driven application. I am going to explain the idea of inferring domain knowledge from ontologies for

- performing computerized tasks
- executing computer orders
- inferring new knowledge and information
- interacting with the user
- interacting with databases and legacy systems

More concretely, we are going to use the ODAM as an adaptor for the implementation of the 13606 dual model approach. This adaptor facilitates the generation of EHR components from archetype descriptions. More about this topic can be found in section 4.3.2.

The second implementation of the ODAM is for automatically interpreting the integrative process model and its individuals. Read more about that in section 4.3.3.

Figure 22 depicts all relevant aspects of ontology driven applications, so I will describe it rather extensively.



**Figure 22: Ontology driven application schema**

Figure 22 depicts the relations between the building blocks of an ontology driven application. The building blocks are shown in a matrix consistent of 2 rows and 2 columns. The left column symbolizes the model level. The first row of the left column shows the abstract ontology model of a domain.

The second row of the model column shows the application site of the ontology model. This is a platform dependent or programming language specific implementation of the ontology model that translates the platform independent ontology model (e.g. provided in OWL) into a programming language structure (e.g. classes and relations or database schema).

The instantiation of the ontology model - as illustrated in Figure 22 in the first row in the instance column – represents a concrete instantiation of a domain model. But what does an instance of a domain model mean exactly? Instances of a domain model are concrete information and knowledge about a specific domain. Whereas the domain model formulates (only) the abstract concepts and properties, the instances refer to the concrete items and elements that really exist in the domain. The ontology instances therefore form a description of real items based on a model written in a formal language. The ontology model and the ontology instances constitute a complete ontology.

The last unexplained cell of the matrix in Figure 22 is the intersection of the instance column and the application row. This cell illustrates the aim of the ontology driven application. Dependent on the application model and on the ontology instances an application derives its functionality. The illustration conceivably predetermines that ontology driven applications result in executable program statements. In fact the results of an ontology driven application depend on the requirements, the application is designed for.

Important prerequisites for these considerations are:

- the ontology model has to be unambiguous
- the ontology model has to be stable (no further developments concerning the ontology model)



### **4.3.2 ISO 13606 and the ontology driven application model**

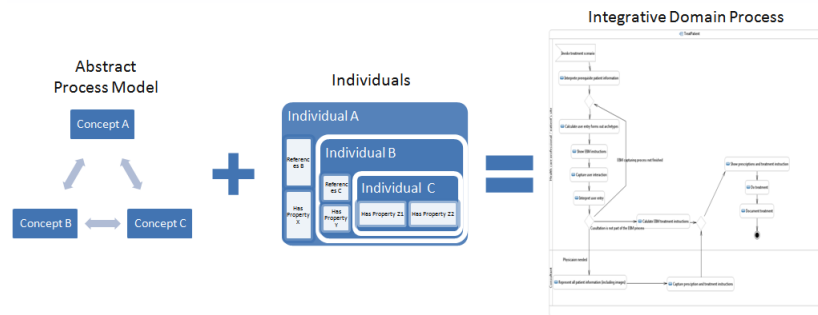
As the dual model approach of the ISO 13606 EHR communication standard has its origin in structuring information using domain ontologies (36), archetypes can be seen as descriptions of information items. The archetype concept is similar to ontologies except that ontologies tend to describe real world things (115).

Implementing the ODAM for ISO 13606, the ontology model is given by the ISO 13606 reference model and the ontology instances are described by the ISO 13606 archetypes (see Figure 22). An application can easily adapt to new archetypes without reprogramming the IT infrastructure and back-end systems (36). Through an appropriate application model, EHR information could be exchanged between legacy EHR systems without losing information during transformation from 13606-based data to the data base schemata of the legacy EHR system.

The ODAM provides a method to retrieve data based on the reference model. Further the ODAM enables querying the entire EHR data repository and drilling down onto archetype information later if necessary.

### **4.3.3 Process representation for an ontology driven application model**

Figure 23 shows a process model that defines the building blocks of a process. These building blocks are for example actions, control flows, decision nodes etc. The individuals describe how to arrange these building blocks to define a concrete domain process. Both, together result in an integrative-domain-process.



**Figure 23: Integrative domain process**

In this work, the abstract process model is based on OWL-S classes and properties. The individuals are instances of these OWL-S classes and properties. Together they form an ontology. By formulating different individuals, several diverse process descriptions based on the same OWL-S classes are possible. Again the ontology driven application schema can be applied on this process descriptions to facilitate:

- the exchange of individuals between different systems
- and to integrate the abstract process model into concrete IT systems, to enable process support and service interoperability

Service interoperability can be reached by communicating individuals and assembling them to an integrative process as shown in Figure 23. Thus several new processes for different contents could be easily implemented without reprogramming the IT infrastructure. Furthermore interchanging individuals allows transmitting information about the state during a process. This means in a real world scenario for example information about how the patient has been already treated and how he has to be further treated. If you imagine a scenario where several attending physicians have to work together and exchange information concerning one patient, process information could be a major benefit.

This approach facilitates the definition of EBM treatment or decision supporting processes for an interactive IT support for health care professionals (HCP). The EBM and GCP knowledge would be constructed as process, the input and output information those trigger the states are archetyped 13606 EHR extracts.

#### **4.4 The wound management process and the wound management archetypes**

In this section I describe the knowledge compounds that back the actual implementation of the wound management process. I also describe the involved archetypes:

- Wound management (WM) general patient info archetype
- WM wound specific archetype
- WM prescription archetype and
- WM treatment archetype

The WM archetypes describe the different required information structures for capturing anamnesis/status -, prescription – and treatment information of patients with chronic wounds.

The structures of the WM archetypes are derived from currently used documentation forms for in-house and out patients with chronic wounds at the General Hospital of Vienna. The structures of the WM archetypes were built in cooperation with the Department of Dermatology at the General Hospital Vienna. The status of WM archetype so far can be seen as draft or experimental. In the next 2 sections I will describe the WM process ontology first (sections 4.4.1 and 4.4.2). Subsequently I will show the domain concepts (archetypes).

##### **4.4.1 The GLIF compatible process-ontology**

As GLIF3 provides an approved and comprehensive methodology for modeling evidence based clinical guidelines electronically, my primary goal is to reuse this methodology within distributed environments and for embedding domain concepts (like archetypes) and electronic health records. GLIF3 tends to model domain knowledge and domain concepts implicitly as part of the electronic guideline knowledge. For example a guideline for treating hypertension uses the values of systolic blood pressure and diastolic blood pressure implicitly without modeling or accessing them as domain concepts exclusively. My aim is to provide service and organizational interoperability by defining common business processes and upper level ontologies (see Figure 41). A common business process or upper level ontology is for example a disease management process. This disease management process should be based on the best available evidence (EBM). For defining EBM processes and guidelines GLIF3 is an approved methodology. But GLIF3 does not support standardized access of EHR data, nor does GLIF3 make use of the domain concepts (e.g. modeled via archetypes), nor was GLIF3 intended to be executed at distributed systems.

The aim of OWL-S, however, is to facilitate service and organizational interoperability. OWL-S has a far more general focus and OWL-S was not intended to be developed for the clinical or medical sector exclusively. Because of the features of OWL-S like service interoperability and semantically describing services I will adapt the OWL-S model to meet the criteria of formulating EBM processes. Therefore I will use the GLIF3 methodology to develop an extended OWL-S model intended for medical and clinical processes.

The next sub-sections refer to the subsections of 3.1 to outline the extensions of the OWL-S model.

OWL-S is not intended to model medical processes exclusively. Therefore it comes with a far more general model, defining single actions (or interactions) called “atomic processes”, and “composite processes”. The composite processes define control structures like a sequence, a split, a split+join, choice, if-then-else, iterated, while and until loops. A composite process spans a tree of control structures. At the leaves of the tree are invocations of other processes (atomic process, or a composite process in case of a nested process). When a process is specified, input variables and preconditions define where the inputs of the process come from and output variables and results define where the outputs go to.

#### **4.4.1.1 Action step**

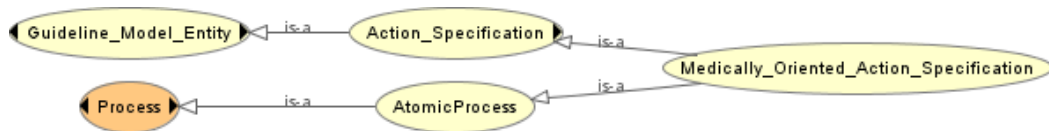
The GLIF3 model specifies different action steps that can occur in an electronic guideline flow chart. The following action steps are among these action step specifications (see section 3.1.1):

- The medically oriented actions, like guideline recommendations;
- The get data object action, an action for retrieving data from an user entry or an electronic medical record (EMR);
- The sub-guideline action, which refers to a nested guideline.

##### **4.4.1.1.1 Medically oriented actions**

A medically oriented action can be seen as specialization of an OWL-S atomic process. Therefore I reused the GLIF3.5 specification to build a subclass of atomic process. The subclass of atomic process complies with the GLIF3.5’s medically oriented action specification. The individuals of this specialization of an atomic process define a clinical recommendation according to the GLIF3.5 specification. OWL supports multi-inheritance. The upper path in Figure 24 shows an excerpt of the inheritance hierarchy of the GLIF3.5. The lower path shows an excerpt of the

process model of OWL-S. For reusing the features of the GLIF3.5 model, I extended the OWL-S ontology with parts of the GLIF 3.5 model. The whole path is an excerpt of the extended OWL-S ontology.



**Figure 24: Medical oriented action specification in an excerpt of the extended OWL-S ontology**

#### 4.4.1.1.2 Get data object action

Obtaining a data value from an EHR is done by calling the web service that provides the access to the EHR. Web services allow to obtain ISO 13606 archetyped data and archetypes. OWL-S facilitates to define and describe service interaction and upper level contexts to describe the usage of the underlying web services. Because of this separation between web services and descriptive logic provided by OWL-S the get data object action is provided by calling the web service for the described archetyped EHR component.

#### 4.4.1.1.3 Sub-guideline action

In GLIF3.5 a sub-guideline action is a high level action that refers to a nested (sub) guideline. Using OWL-S a guideline is represented as a composite process. Composite processes can be nested. Composite processes consist of control structures which span a tree. Atomic processes form the leaves of the tree performing any kind of interaction, or composite processes representing nested complex processes. For expressing a sub-guideline action as defined in GLIF3.5 a nested composite process has to be defined using OWL-S.

#### 4.4.1.2 Decision step

GLIF3.5 includes four different models of decision steps. The first type of decision steps models deterministic decisions, and the other types of decision steps model non-deterministic decisions (see section 3.1.2). GLIF3.5 models all decision step classes by accessing implicit domain knowledge. In other words GLIF3.5 models domain knowledge as decision steps. A decision step in GLIF3.5 is not just an element of control flow; a decision step also models the semantic of the decision options.

Using ISO 13606 compliant web services accessing EHRs, and using OWL-S for semantically describing the relations between these web services follows another approach than GLIF3.5. Data communication between EHRs and an OWL-S process description are performed by interpreting input and output variables in a control structure. These input and output variables can refer to elements in an EHR. The semantic of the GLIF3.5's decision option however is not modeled by the control structure of the OWL-S process but within the domain concept that belongs to the EHR. The control structure of the OWL-S process defines the expression for evaluating the condition of an EHR (or data items within it).

The domain concept defines the semantic of the decision options for deterministic decisions, non-deterministic rule based decisions and non-deterministic weighted choice decisions. For example a blood pressure domain concept (archetype) defines an item for "level of BP"; the archetype also has to define the decision options like "presence of renal insufficiency", "presence of diabetes and no renal insufficiency", "presence of cardiovascular disease insufficiency or diabetes". The corresponding OWL-S control structure - that accesses the archetype based EHR - evaluates the value of this data item (the value is set, because the EHR provides the patient information) and performs the next process according to the evaluation expression. In this example the OWL-S control structure evalu-

ates to one of the three action steps each representing a recommendation (one action step for each decision option of the data item).

Utility choices are not supported so far.

#### **4.4.1.3 Branch step**

In OWL-S, a branch can be modeled using the “split” control structure. The split control structure facilitates parallel execution of processes.

#### **4.4.1.4 Synchronization step**

The “split+join” control structure of OWL-S allows to define parallel processes which have to be completed before the control flow can proceed.

#### **4.4.1.5 Patient state step**

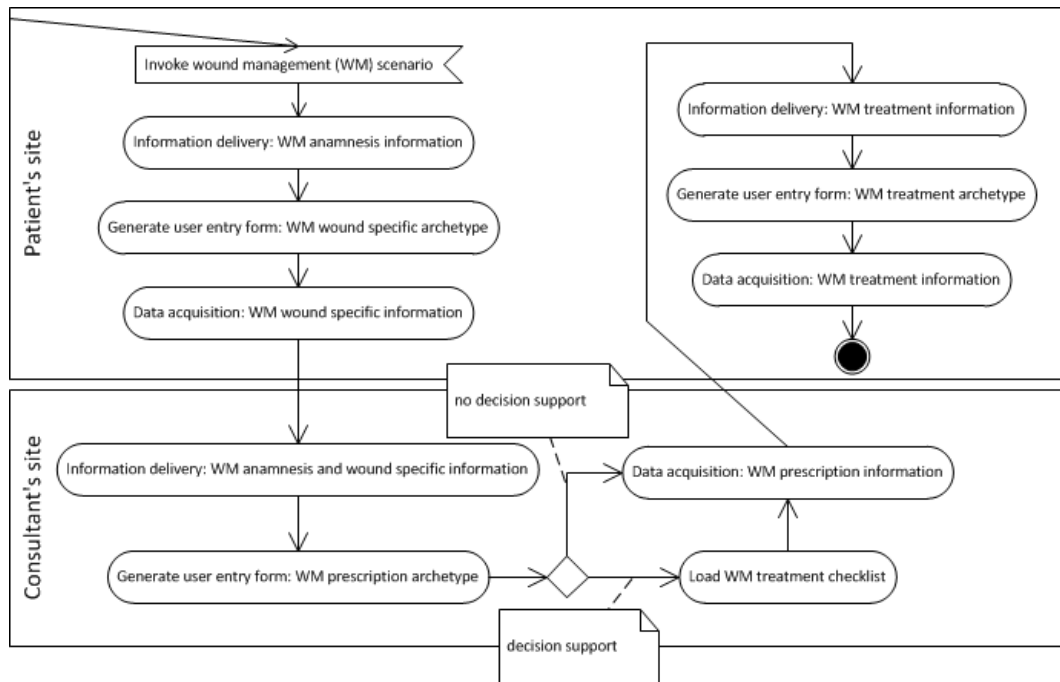
A patient state is typically provided by the EHR data. A patient state step therefore is represented as a web service call performed on an EHR and evaluated by an “if-then-else” control structure of the OWL-S process. A patient state in GLIF3.5 is also used to find the starting point within a guideline. An entry point within the OWL-S process can be modeled using ISO 13606 EHRs by evaluating the preconditions and the input variables which lead to an execution of a certain process within the composite process. Using ISO 13606 compliant EHRs, the precondition of an OWL-S process is the archetype (e.g. archetype id). If an archetype (domain concept) is used more than once within an OWL-S process, the EHR item definition within an archetype is used additionally to identify the state of the process. For example the execution of an OWL-S control construct reads the archetype id of an EHR that has been communicated for a certain process support. The execution jumps to the atomic process (state) whose input is associated



with the archetype, reads the corresponding EHR patient data and evaluates the expression of the control construct to finally call the evaluated next process (e.g. a medical recommendation as atomic process).

#### 4.4.2 The wound management process

Figure 25 shows the actual wound management process that is implemented to support physicians and wound manager to enable a multi-professional homecare management.



**Figure 25: wound management process (UML activity diagram)**

The wound management process as shown in Figure 25 illustrates the wound management scenario only. This process does not cover precedent default processes that are environmental for the wound management scenario. Such processes are for example login and authentication, patient search and anamnesis documentation (prerequisite information captured during inclusion procedure at the hospital or consultant). These processes are of course part of the application but as they are straight forward concerning their input, process and output, they are not scope of this thesis.

The process depicted in Figure 25 is triggered by an event that passes prerequisite patient information to the process (see Figure 25 “Invoke wound management (WM) scenario”). This preliminary patient information has to be captured during the anamnesis or the admission procedure that leads to a patient’s admission to the multi-professional wound management homecare treatment. This prerequisite patient information is archetype based EHR data that are captured before the process can be performed. Here the prerequisite patient information is EHR extracts based on the wound management archetype. The wound management archetype will be described in section 4.4.3.

The next action illustrated in Figure 25 is the “Information delivery: WM anamnesis information” event. Here the just mentioned prerequisite-archetype-based-EHR-patient information is being delivered and shown in the user interface.

During the next action “Generate user entry form: WM wound specific archetype”, the wound management archetype for a specific wound is being loaded and a user entry form is being represented to the user. This user entry form is used to document patient’s status information.

After and during the nurse documents the patient information, the captured data are interpreted and validated (Figure 25 “Data acquisition: WM wound specific information”). The scope of an archetype is to formally describe the structure of the data that should be communicated. Based on this description, validation rules are derived which are evaluated during this action before an EHR extract is communicated. The output of this action is an EHR extract.

The next action (Figure 25 is the “Information delivery: WM anamnesis and wound specific information”) is triggered at the consultant’s site where the patient’s anamnesis and wound specific information are represented to the user.

To capture the prescription data, the user interface has to be generated first which happens at the action “Generate user entry form: WM prescription archetype” as depicted in Figure 25. The user interface generated from the prescription archetype provides a decision support event.

- Case 1: If the consultant triggers the decision support event the conditions of a sub-process (see Figure 26) are evaluated and the recommendations (see Figure 26) are generated. The consultant should agree to the generated recommendations (e.g. by clicking on a button) before he/she transfers the automatically generated values to the prescription documentation.
- Case 2: if the consultant does not trigger the decision support event, the recommendations are not generated.

In the next action (Figure 25 “Data acquisition: WM prescription information”) the consultant captures the prescription information and/or edits the recommended values transferred from the decision support.

On the patients site the nurse receives the treatment instructions. The treatment instructions are represented to the nurse (see Figure 25 “Information delivery: WM treatment information”). Then the patient is being treated.

Subsequently a user entry form is being generated, based on the wound management treatment archetype to capture the treatment information (see Figure 25 “Generate user entry form: WM treatment archetype”).

The last action in this process (see Figure 25 “Data acquisition: WM treatment information”) is to capture the actual information about the patient’s treatment (e.g. which bandage was used) and send this information as EHR extract to the server which stores the extracts.

At the node “Load WM treatment checklist” (see Figure 25) a sub-process is triggered. This sub-process (see Figure 26) provides decision support for assigning the appropriate treatment, which is derived from best available evidence for the treatment of chronic wounds, summarized in (4).

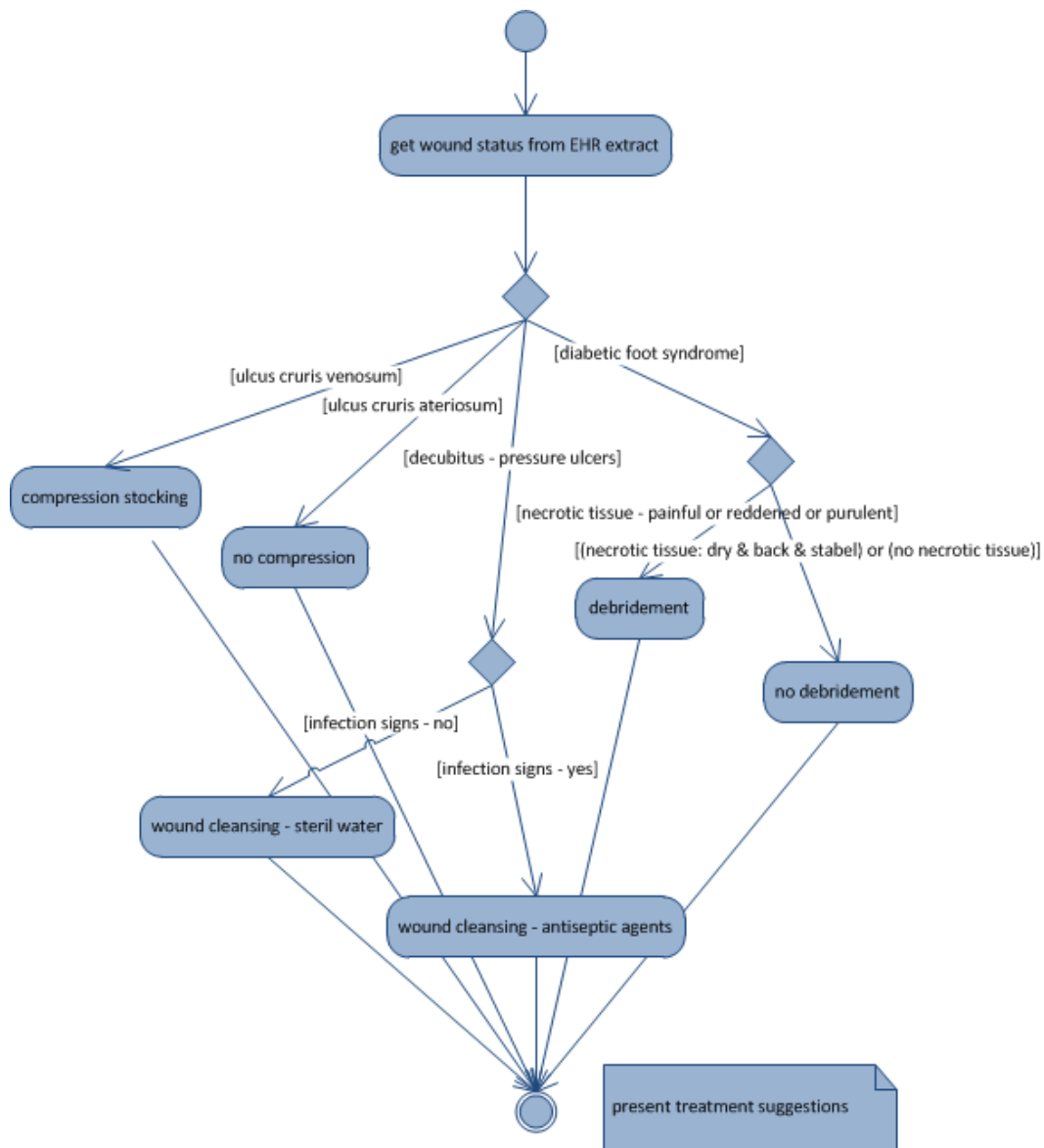


Figure 26: assignment of treatment options

The wound management process, (Figure 25) and its sub-process (Figure 26) are modeled by the individuals of the extended OWL-S ontology using RDF/XML syntax.

#### **4.4.3 Wound management general patient info archetype**

Figure 27 shows the structure of the general information of a patient with chronic wounds. An EHR\_EXTRACT based on this archetype structure is intended to be captured once for each patient. The domain information in this archetype covers the chronic diseases underlying the patient's chronic wound(s) and the influencing factors which may also cause bad healing wounds. The patient information of both the primary disease and the influencing factors are not likely to change very often. Therefore we separated this domain information in the general wound management patient info archetype so that the users do not have to capture this information at each visit. An EHR\_EXTRACT containing information of this domain will be created at the clinic before the patient is admitted to the multi professional homecare management.

The form in Figure 27 is based on a COMPOSITION archetype. This means that the highest level class of the reference model constrained and structured by the archetype is a COMPOSITION. Furthermore a COMPOSITION archetype can directly be used to generate a new corresponding record within an EHR\_EXTRACT. Primary disease and influencing factors are structured as SECTIONS. Within these SECTIONS there are the ENTRIES (like weight, height, occlusive artery disease, etc.) with their ELEMENTs and assumed values (like yes, no, I, II, III, etc.) see Figure 27.

**Wound management general patient info archetype**

**Primary disease**

Weight  kg      Height  cm

Occlusive Artery Disease 

- not available
- yes
- no
- I (Fontaine)
- II (Fontaine)
- III (Fontaine)
- IV (Fontaine)

Diabetes Mellitus 

- not available
- yes
- no
- type I
- type II

CVI 

- not available
- yes
- no
- I (Widmer)
- II (Widmer)
- III (Widmer)

Pressure 

- not available
- yes
- no

Trauma 

- not available
- yes
- no

Immobility 

- not available
- yes
- no

**Influencing factors**

Nicotine Abuse 

- not available
- yes
- no

Malnutrition 

- not available
- yes
- no

Reduced General Condition 

- not available
- yes
- no

Metabolic Disorder 

- not available
- yes
- no

Cortisone Taking 

- not available
- yes
- no

Immunosuppression 

- not available
- yes
- no

Infectious Diseases

Comments

**Figure 27: structure of general wound management patient info form (see section 8.1.1 for the corresponding ADL-format)**

#### **4.4.4 Wound management archetype – wound specific**

Figure 28 shows the form that is used to collect record components of an EHR\_EXTRACT which document the wound specific patient information. Wound specific information may change frequently and must be documented for every treatment and patient visit. A frequent documentation is necessary to keep track of the wound healing process and to plan further treatment. A patient often has more than one bad healing wound at different locations therefore it is necessary to document each wound separately. A record component - of an EHR\_EXTRACT - containing information of this domain will be created at the beginning of the patient's treatment process. Assuming that the patient is admitted to the multi professional homecare management process, the HCP will capture the patient information of this (sub) domain at the patient's site.

This archetype is a COMPOSITION archetype containing one ENTRY. This ENTRY structures the wound specific information (see Figure 28). When generating a record component for an EHR\_EXTRACT based on this archetype this ENTRY is intended to be captured repeatedly to document more than one wound. The localization (location of the wound) ELEMENT is especially important because it identifies a wound. The localization of the wound is used later to assign the wound specific treatment.



Wound management archetype (wound specific patient info)


**Wound Specific Info**

Localization

Wound

ulcus cruris	decubitus	diabetic foot syndrome
ulcus cruris arteriosum	decubitus grade I	postoperative impaired wound healing
ulcus cruris venosum	decubitus grade II	other: Text <input type="text"/>
ulcus cruris mixtum	decubitus grade III	
	decubitus grade IV	

Duration  Recurrence count

Wound Images  Length  cm  
Width  cm  
Depth  cm

**Reason**

granulating  
 epithelializing  
 fibrin  
 necrotic  
 Other

**Border**

bland  
 reddened  
 edematous  
 maceratus  
 epithelializing  
 Other

**Surrounding**

bland  
 reddened  
 dry  
 edematous  
 maceratus  
 livid  
 hyperthermic  
 Other

Healing stage

cleaning
granulation tissue formation
epithelialization

Exudate

none
little
moderate
a lot

Woundsmell

not available
yes
no

serous  
 sanguineous  
 purulent  
 greenish  
 Other

Pain

not Available
no
yes 0 (little)
yes 1
yes 2
yes 3
yes 4
yes 5
yes 6
yes 7
yes 8
yes 9
yes 10 (extreme)

Signs of infection

lymphangitis  
 swelling of the lymph node  
 fever  
 Other

Comments

Figure 28: structure of the wound specific patient info form (see section 8.1.2 for the corresponding archetype in ADL-format)

#### **4.4.5 Wound management prescription archetype – wound specific**

Figure 29 shows the form that is used to collect record components of an EHR\_EXTRACT which document the assigned treatment for each wound. Assuming the multi professional homecare management scenario, the information of this domain is captured by the tele physician at the clinic or physician's office. A patient often has more than one bad healing wound therefore a documentation of the prescription of each wound is necessary.

The wound management prescription archetype is a COMPOSITION archetype containing one ENTRY, "wound specific treatment". When capturing a record component for an EHR\_EXTRACT based on this archetype this ENTRY may be captured repeatedly to document the intended treatment for each wound.

This archetype is nested, i.e. the ENTRY "wound specific treatment" is the same ENTRY as in the wound management treatment archetype (see section 4.4.6). The wound specific treatment ENTRY is defined separately in a ENTRY archetype. Using it within the prescription archetype facilitates to share the same structure of information also used for structuring the WM treatment archetype. The only distinction - why to use different upper-level archetypes (prescription vs. treatment archetypes) - is because they fulfill different domain purposes, but share the same structure of information.

Wound management prescription archetype (wound specific prescription)

**Wound Specific Treatment**

Localization

Frequency of Treatment  daily   ... times a week

Wound Cleansing

Debridement

Wound Border Protection

Wound Specific Medication

Compression

short bandage
stocking
other: (text)

Comments

Figure 29: structure of the wound specific prescription form (see section 8.1.3 for the corresponding archetype in ADL-format)

#### **4.4.6 The wound management treatment archetype – wound specific**

Figure 30 shows the form that is used to collect record components of an EHR\_EXTRACT which document the treatment of the patient. The treatment of a bad healing wound is performed specifically for each wound. Therefore also the treatment documentation has to be specific for each wound. Assuming the multi professional homecare scenario, the treatment and the documentation of the treatment are done by the HCP at the patient's site. The prescription information has been already recorded by the physician. To facilitate electronic data capturing process the prescription information can be preloaded to accelerate the documentation process. Furthermore it is very likely that the HCP follows the prescription of the physician but it has to be documented that the treatment was performed.

The wound management treatment archetype is a COMPOSITION archetype containing one ENTRY "wound specific treatment" (see Figure 30). When capturing a record component for an EHR\_EXTRACT, based on this archetype, this ENTRY may be captured repeatedly to document the performed treatment for each wound.

This archetype shares a nested section archetype with the prescription archetype above. Sharing the same structure of information makes it easier to preload the physician's prescription information.

Wound management treatment archetype (wound specific treatment)

**Wound Specific Treatment**

Localization

Frequency of Treatment  daily   ... times a week

Wound Cleansing

Debridement

Wound Border Protection

Wound Specific Medication

Compression

- 
- 
- 

Comments

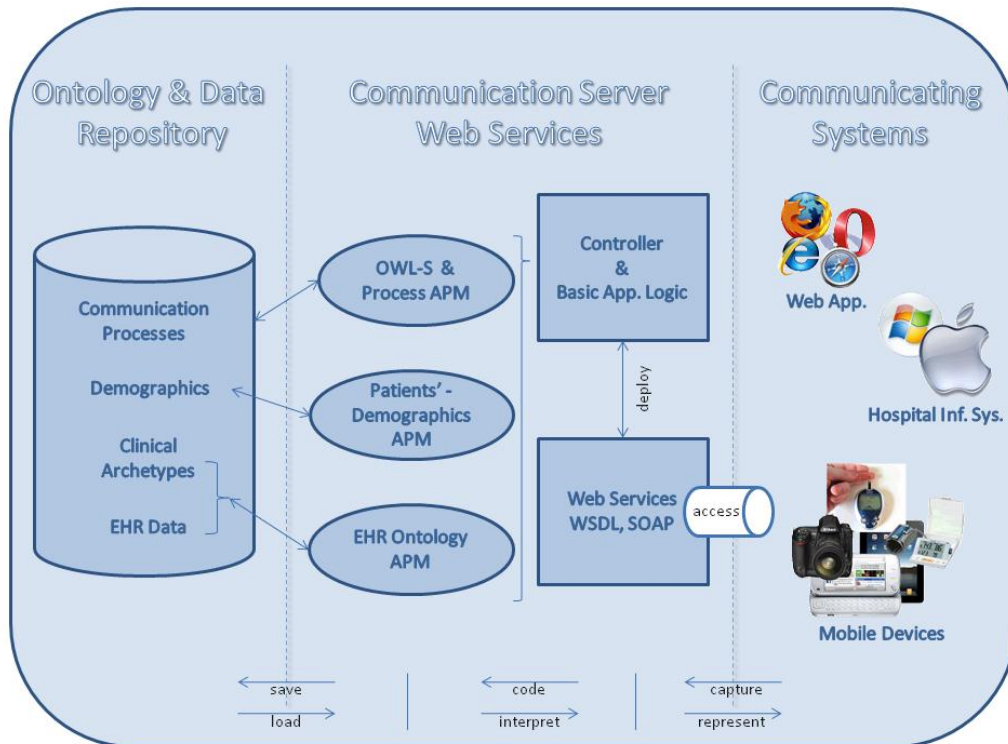
Figure 30: structure of the wound specific treatment form (see section 8.1.3 for the corresponding archetype in ADL-format)

## 4.5 System architecture

In this section I will describe the proposed system architecture for applying evidence based tele wound management in a distributed semantic web environment. First I will show and describe the major building blocks of the application architecture. Then I will depict some delimitations which are not part of the implementation yet. And finally in this section I will outline the interface to access the functionality of the system.

Figure 31 illustrates the system architecture, of an ontology driven application for a distributed environment. It consists of three columns.

- Ontology and data repository
- Communication server and web services
- Communicating systems



**Figure 31: prototype system architecture**

Let us have a closer look onto these areas:

#### **4.5.1 Ontologies and data repositories**

The left column illustrates the persistent data layer; here all application relevant data are stored and retrieved. The data are stored as individuals of an ontology. For this purpose tree different ontology repositories are implemented.

The first one holds the individuals of the process description, which defines the communication process (73). This is for example the wound management process shown in section 4.4.2. I use OWL-S to describe the process semantically. For more details about OWL-S see section 2.5.

The second repository is the patients' demographics ontology. Its individuals identify the subjects.

The third ontology repository is for persisting medical information associated with patients. This information is also stored as an ontology. As shown in section 5.3, I decided to internally represent the ISO 13606 standard as an OWL ontology. This ontology representation covers the information about the used archetypes as well as the EHR extract information. Archetypes and EHR extract information are represented as individuals within this ontology.

#### **4.5.2 Communication server and web services**

The middle tier of the system architecture depicted in Figure 31 focuses on the web services that accesses the ontology repositories, reveals the ontologies' semantics and grants access to the system. Let us have a closer look onto the Application Models (APM see section 4.3.1) on the left side surrounded by ellipses.

##### **4.5.2.1 OWL-S and process application model**

The first APM (OWL-S & Process APM) provides access to the process ontology. Its primary functionality is to grant access to the process ontology.

The secondary functionality of this APM and its corresponding ontology is to describe and query the relations to the patients demographics ontology. As the subjects demographics ontology captures the patient identifying information the OWL-S & Process APM reveals which process relates to the patient.

##### **4.5.2.2 Patients - demographics application model**

The second APM (Patients - Demographics APM) is derived from the subject demographics ontology and developed to save, load and interpret patient identify-



ing information (like name, address, insurance number, etc.). As the subject's demographic ontology also describes the associations with the EHR extracts and the (wound management) processes, this APM's scope is also to query the associations to the medical contents (links to the EHR extracts and processes for which the patient is registered). This means, committing a query onto a certain patient, the APM returns the identification of this patient. Using this identification allows querying onto EHR extracts. If new information has been captured, this APM provides the functionality to store or update the patient identifying information as well as the associations to the patient's clinical contents (EHR extracts and/or processes).

#### **4.5.2.3 EHR ontology application model**

The functionality of the APM called "EHR Ontology APM" is to access the EHR ontology (see section 4.5.4.2 for an overview) that has been built on the principles mentioned in section 5.3. ISO 13606 and the archetype model are basically two models described through UML. Instantiations of one of these models are either an EHR extract or an archetype. The latter may also be specified by means of the ADL, and this can be seen as instantiation instruction for the archetype model. The EHR ontology represents the 2 models as one OWL ontology. For basic information and related work see section 5.3. As the expressiveness and the grade of abstraction achievable by means of ontologies are higher than with object oriented methods, I joined both models (reference model + archetype model) to one ontology, the EHR ontology. The aim of the work described in section 5.3 was to build an OWL representation for archetypes. However to load, store and interpret EHR extracts I had to extend this ontology to meet the requirements of ISO 13606 Part 1. The archetype ontology (the union of the 2 ontologies mentioned in section 5.3) already describes all concepts of ISO 13606 Part 1 except EHR\_EXTRACT and EXTRACT\_CRITERIA. This means that I had to

add the concepts (classes and properties) of EHR\_EXTRACT, and EXTRACT\_CRITERIA to the archetype ontology, to create an ontology (EHR ontology) capable of meeting the criteria of conformity to 13606 Part 1 and capable of representing EHR\_EXTRACTs as OWL individuals.

Individuals of this ontology are EHR extracts and archetypes. Conformant to the ISO 13606 standard, EHR extracts can exist without archetypes, or an archetype can annotate and describe an EHR extract.

The APM allows to access the information of the individuals, and to compute this information programmatically. It covers the major ontology driven functionality for creating EHR extract components on the definition and constraints of an archetype. The functionality of the APM is also to create new EHR extracts to store new patient information in the ontology. This APM can be seen as wrapper for all the 13606 relevant operations as it is called by the APM – “OWL-S & Process API” (see section 4.5.2.1).

**Limitations of the prototype implementation:** The ISO 13606 web services do not access the EHR ontology at the current development state of the prototype implementation. The prototype implements an object oriented database to query and store the EHR\_EXTRACTs. The web service structure and the web service messages are based on the class structure that is also used to store the EHR\_EXTRACTs in the object database. The class structure is a simplified structure of the ISO 13606 reference model that implements all required attributes of the reference model.

#### **4.5.2.4 Controller and basic application logic**

The functionality of this server side module is to provide the basic application - and controller logic, to call the APMs and pass information from/to the web services. The basic functionalities are:

- User authentication
- User administration (register, change password, update user information)
- Patient search
- Patient documentation call (load, save EHR extracts, process insensitive)
- Process call (calls the process scenario, loads and saves EHR extracts, generates working list for pending processes)
- Logout

#### **4.5.2.5 Web services**

The web services provide the access to the server functionality. There are two categories of web services:

- Basic application logic.
- General APM logic.

The basic application logic web services provide the functionalities listed before in section 4.5.2.4.

The general logic web services are mainly embedded in and accessed by the basic application logic - web services. This means that a communicating system first calls a basic controller logic web service (e.g. to guarantee authentication). Subsequently a controller web service calls a functionality of the generic APM logic. Web services are provided to access the OWL-S process APM and the EHR ontology APM.

The web services for the OWL-S process APM grant access to the process ontology and return the ontology and its individuals to be interpreted at the calling system. The process definition is provided as individuals of the modified OWL-S ontology.

The web services for the 13606 reference model and archetype model are provided, to support semantic interoperability of the 13606. These web services enable client applications to communicate EHR extract data to the server. They also facilitate querying archetypes and provide instances of the archetype object model (archetypes via WSDL/SOAP).

Figure 32 illustrates the generic APM logic accessed through web services.

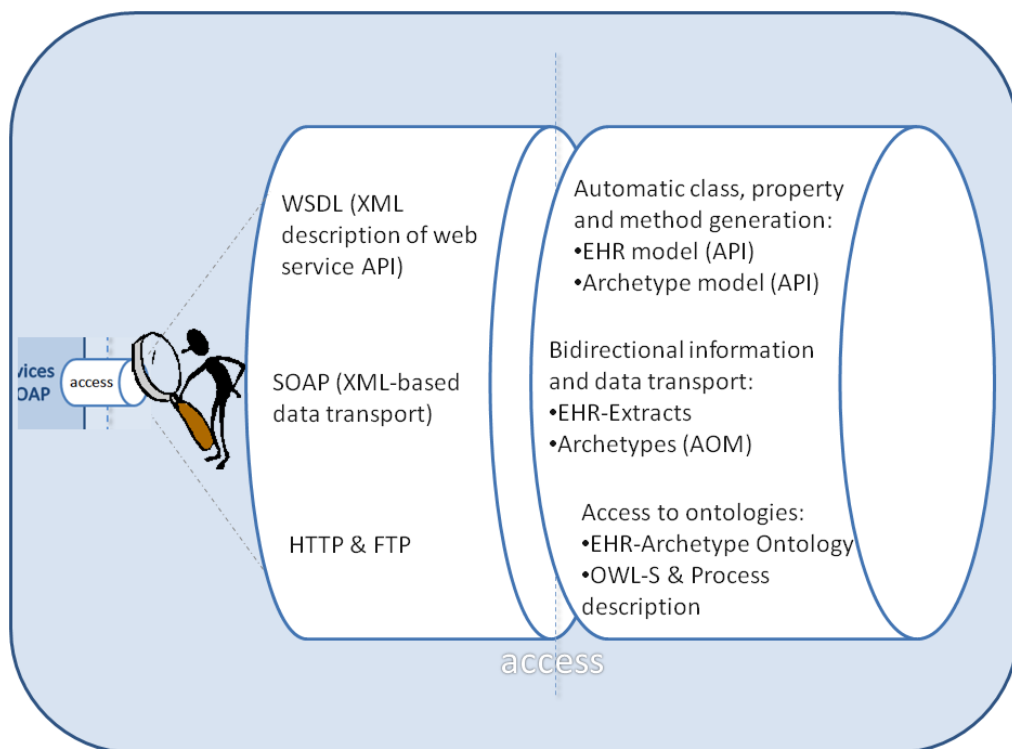


Figure 32: prototype system architecture (access) – extract of Figure 31

### 4.5.3 Communicating systems

The communicating systems communicate with each other through the web services of the communication server. The prototype implements a web application which is called Ontology Driven EHR Web (ODEWeb). This ODEWeb accesses the web services and exchanges data through the provided interfaces. The web application is a reference implementation to show the functionality of the web service calls. ODEWeb implements an APM to automatically interpret the extended OWL-S ontology and the wound management communication process. Furthermore ODEWeb derives the functionality for suggesting decisions from the extended OWL-S ontology by evaluating the decision tree shown in Figure 26.

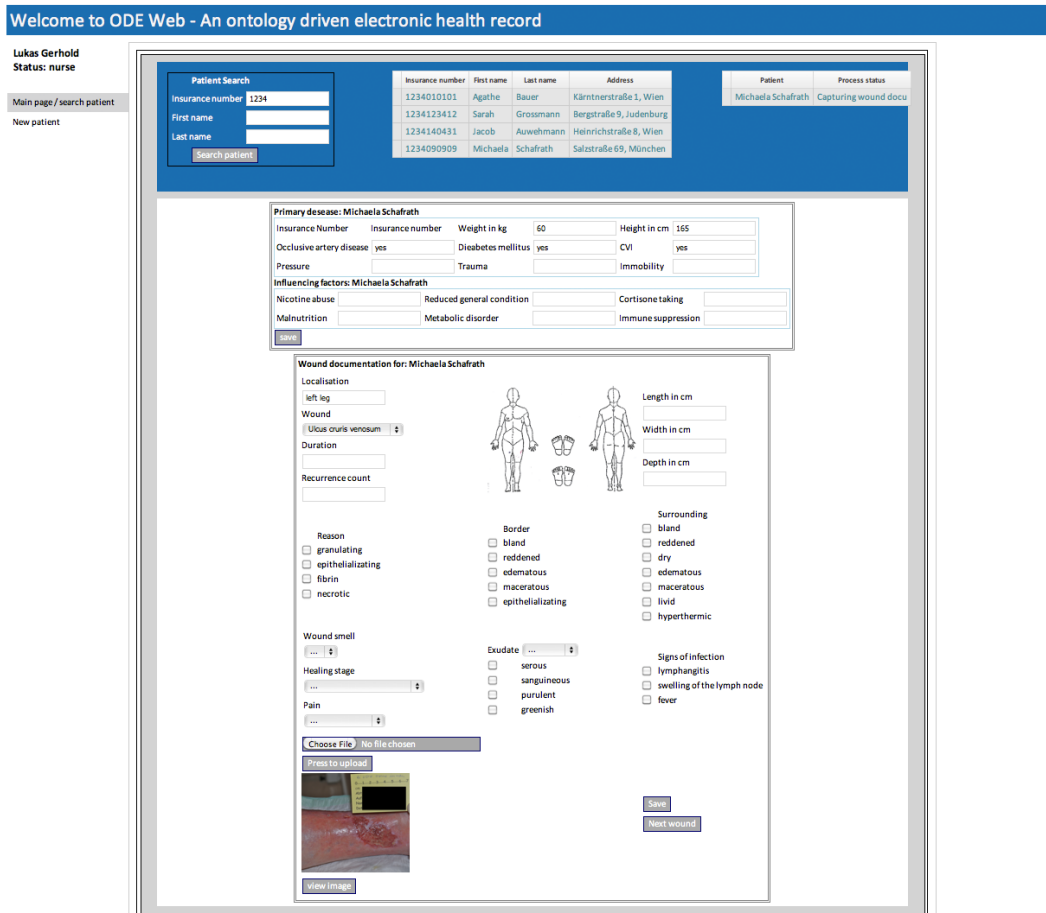


Figure 33: ODEWeb - wound documentation - view of the nurse

Figure 33 shows the ODEWeb application in the nurse view to capture the patient's wound documentation. After completing the wound documentation the patient information are communicated via an EHR\_EXTRACT to the communication server. To continue the communication process a physician has to access the communication server for retrieving the EHR\_EXTRACT containing the wound documentation.

Figure 34 shows the entrance page of another system called Dermatrials which is used at the Department of Dermatology at the General Hospital of Vienna to make web based clinical trials. For demonstrating the distributed access to the communication server, Dermatrials also implements the web services for communicating the EHRs. Figure 34 depicts the pending wound documentation in the box at the bottom of the entrance page.

The screenshot shows the Dermatrials application interface. At the top left is the logo of the Medical University of Vienna. At the top right is the logo of the Division of General Dermatology, Department of Dermatology. Below the logos, it says "OD EHR logged in as: Markus Dorn – Angemeldet als: M D".

The main content area contains two tables:

Studivnummer	Bezeichnung	Review	Patienteninformationen	Studiendefinition	Patientenliste
2	Test	zuerst Patienteninformationen erfassen	erfassen	abgeschlossen	anzeigen


OD-EHR Wunddokumentation	Patient (SVNR)	Status	Aktion
	1234090909	Wounddoku abgeschlossen	Verschreibung erfassen

**Figure 34: Dermatrials - overview showing the pending wound documentation**


Figure 35 shows the physician's view in Dermatrials. This view is the next step of the communicating process and provides the patient information from the EHR\_EXTRACT. It shows the patient information that has been previously captured by the nurse in the ODEWeb application. The physician captures the prescription information and subsequently sends this information as EHR\_EXTRACT

back to the communication server. With this the communication process ends at the physician's site and continuous at the nurse's site.

Figure 36 shows the proceeding of the communication process at the nurse's site. The patient information and the prescription information are displayed. If the nurse clicks on the "treatment done" button the patient's treatment is documented and communicated to the server, and the communication process ends.



**MEDICAL  
UNIVERSITY  
OF VIENNA**



DIVISION OF GENERAL DERMATOLOGY  
DEPARTMENT OF DERMATOLOGY



OD EHR logged in as: Markus Dorn -- Angemeldet als: M D

---

**Wundanamnese(n) - 1234090909**

Caption	Value (version - 0)	Value (version - 1)
Weight	55	60
Height	166	165
Oclusive Artery Disease	yes	yes
Diabetes	yes	yes
CVI		yes
Pressure		
Trauma		
Immobility		

**Aktuelle Wunddokumentation - 1234090909**

Caption	Wound - 0	Wound - 1
Localization	left leg	right leg
Wound	Ulcus cruris venosum	Decubitus grade III
Duration		
Image		
Length		15
Width		6
Depth		1

**Aktuelle Wundverschreibung - 1234090909**

Caption	Wound - 0	Wound - 1
Localization	left leg	right leg
Frequency	5 times a week	7 times a week
Cleansing	yes	yes
Debridement		
BorderProtection		
SpecificMedication		
Compression		

abschicken

**Figure 35: Dermatrials – physician's view – wound prescription**

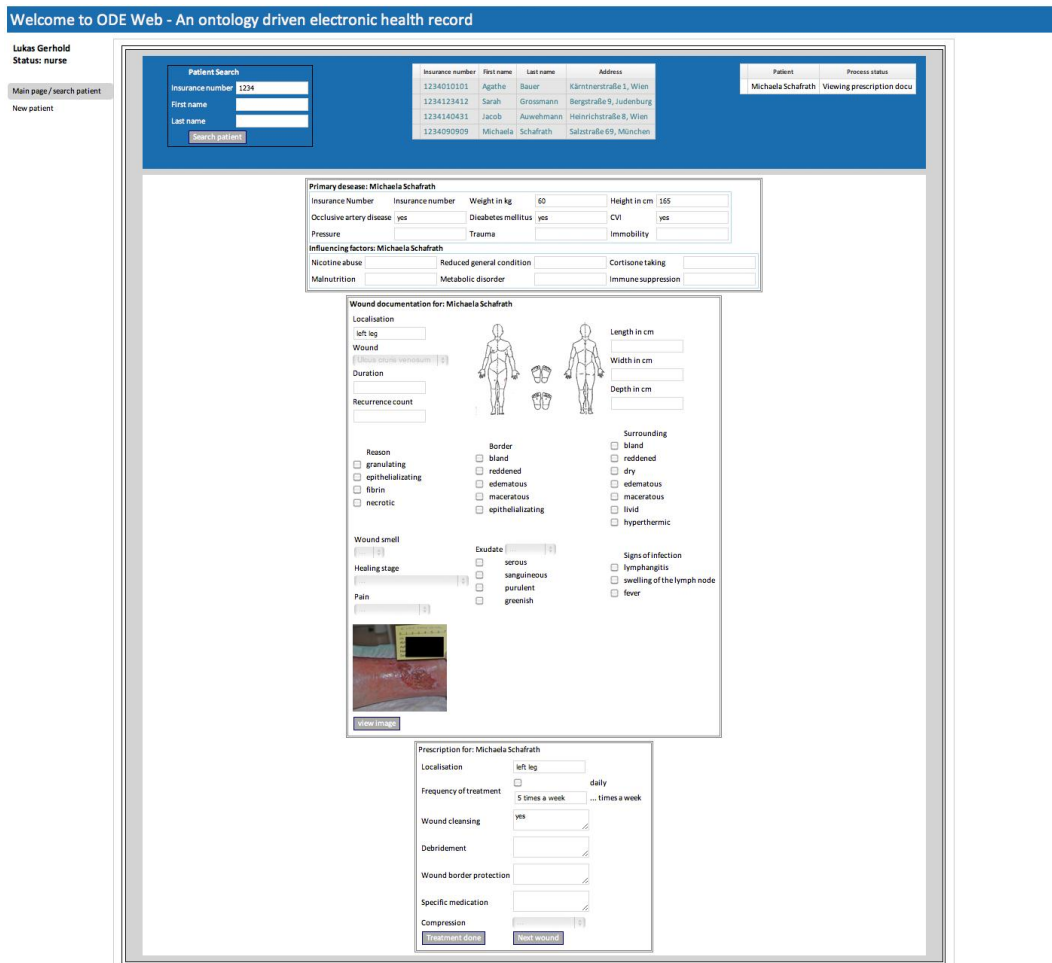


Figure 36: ODEWeb - view of the nurse - prescription view

#### 4.5.4 Ontologies

The ontologies below consist of two levels, the model and the individuals. The model is represented by classes and properties (relations). It reflects the semantic relations abstractly. The individuals represent data entities based on the abstract model. They are persisted in ontology files in RDF/XML syntax. The access to these files is provided by tool kits and APIs described in section 3.4.1.3.



#### 4.5.4.1 Demographics ontology

I developed the demographics ontology model according to the demographics package of the ISO 13606 standard. The demographics ontology serves to identify health care professionals and subjects of care (patients). Both can be referenced by the EHR-ontology.

The individuals of this ontology represent either healthcare professionals or subjects of care (patients). Both are addressed by a unique identification.

Figure 37 provides an overview of the classes in the demographics ontology.

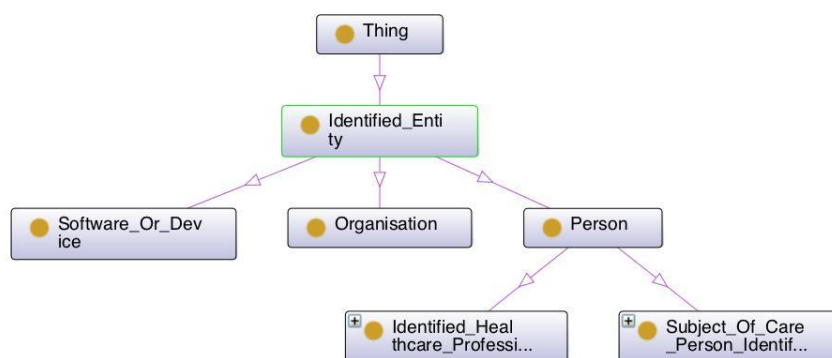


Figure 37: Demographics ontology: OWL-class hierarchy as graph, print made in Protégé

Figure 38 depicts the data type properties of the demographics ontology.

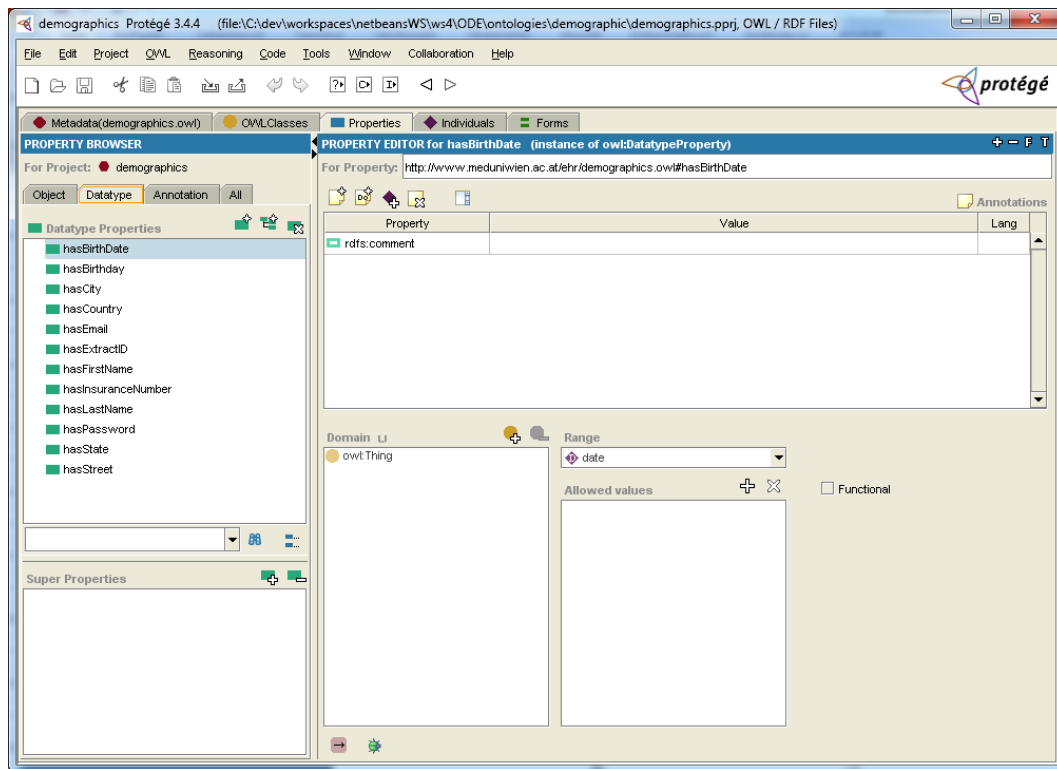


Figure 38: Demographics ontology: data-type overview

The demographics ontology can be downloaded from (116).

#### 4.5.4.2 EHR- ontology

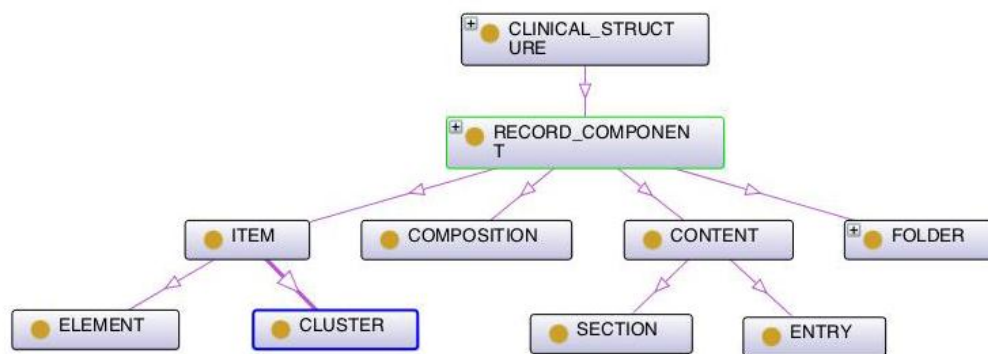
As already mentioned in section 4.5.2.3 the EHR ontology is an ontology representation of the ISO 13606 standard. Based on the work described in section 5.3 we added the EHR Extract specific classes to the archetype ontology.

The result is an ontology that comprises the dual model approach of ISO 13606. The classes and properties of the ISO 13606 reference model are represented as OWL classes and properties. This also applies for the archetype ontology which is described in section 5.3.

The individuals (instances) of the EHR ontology have therefore 2 root individuals. The first root individual is an archetype individual which represents a certain archetype. All individuals that are referenced within an archetype individual represent a certain archetype complying with the instantiation instruction of the ISO 13606 archetype model.

The second root individual corresponds to an EHR extract, keeping all patient related information concerning one patient.

Figure 39 shows the classes of EHR ontology in the Protégé editor.



**Figure 39: EHR ontology: excerpt of the OWL-class hierarchy as graph, print made in Protégé**

Figure 40 provides an excerpt of all properties of the EHR ontology.

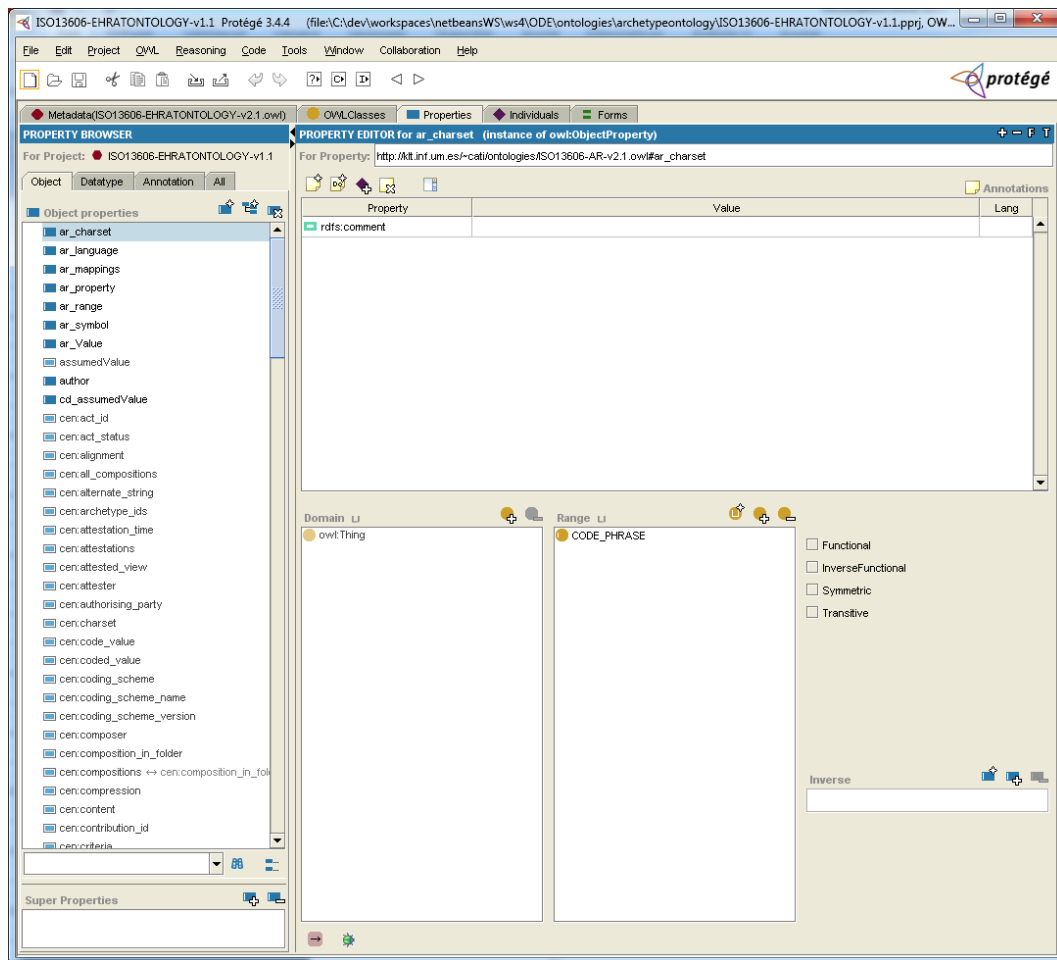


Figure 40: EHR ontology: properties excerpt

The EHR ontology can be downloaded from (117).

#### 4.5.4.3 OWL-S individuals

The structure of OWL-S is already described in section 4.4.2.

The individuals of this ontology describe the web services called for executing the wound management process. The execution instructions are defined as OWL-S individuals.

The web services described in OWL-S the wound management process are:

- wsGetEhrExtract: its functionality is to query an EHR\_EXTRACT with the subject of care identification and to return the EHR\_EXTRACT,
- wsGetEhrExtractRequestNewComposition: its functionality is to query an EHR\_EXTRACT with the subject of care identification, and to return the EHR\_EXTRACT with a new COMPOSITION for a requested archetype,
- wsUpdateEhrExtract: its functionality is to pass an EHR\_EXTRACT to the communication server and to update the content of the EHR\_EXTRACT on the server.

## **5 Discussion and related work**

### **5.1 Wound management**

Wound management comprised the concepts of wound and management. The concept of management means to govern, to lead and to organize. Basically the concept of management describes a systematic process which is performed on different levels. The aim of the management is to find specific solutions for complex problems (4).

A wound is defined as, the separation of different layers of tissue due to external or internal causes. Also deeper structures such as ligaments, tendons or bones can be damaged in addition to the skin layer of the epidermis, dermis or subcutis (118).

From the literature, two main kinds of wounds are known: acute and chronic wounds. Chronic wounds are particular challenges for all participants involved in the wound healing process.

Professional and up-to-date wound management means a systematic multi-professional process which is performed on different levels. It comprises actions

to prevent chronic wounds, actions to accelerate the healing process of chronic wounds, actions to avoid recurrences of chronic wounds and actions to improve the patient's quality of life. The levels of wound management outline the basis of the quality assurance and quality management of diagnosis and treatment of chronic wounds (4).

Lichtenstein mentions in (4) three levels of wound management:

### **Micro level**

The micro level of the wound management describes all patient related actions like diagnosis and treatment. It includes the correct wound diagnostic and the right wound bandage appropriate to the healing progress of the wound (1).

### **Intermediate level**

The intermediate level describes the multidisciplinary action and interactions of all persons (physician, nurse, pharmacist and relative) involved in the treatment of the patient (1).

### **Macro level**

The macro level describes the cooperation between institutions (like hospitals, physician's office, health insurance companies, etc.). The macro level describes the political and economic aspects of the wound management (1).

Wound management comprises all 3 levels. A wound manager is a person whose duty is to assign and perform the correct patient's treatment (at the micro level). Furthermore a wound manger coordinates and involves all persons of each of the multi-disciplines involved in the patient's treatment process (the intermediate

level). And finally a wound manager operates as coordinator between institutions on the macro level (4).

Quality assurance and quality management of diagnosis and treatment of chronic wounds may use ICT to facilitate (4):

- wound diagnostic on the micro level e.g. by implementing telemedical applications to discuss differential diagnoses,
- the communication between all involved disciplines independent of time and location on the intermediate level (e.g. by implementing telemedical support for screening patient's wounds at their home),
- the communication between all institutions involved in the management of patients with chronic wounds.

As the work of Lichtenstein (4) focuses on medical care and the wound management process it does not reflect any IT architecture. Nevertheless Lichtenstein provides a good overview about the actors and institutions of the wound management process. Furthermore Lichtenstein describes in her work the responsibilities and regulatory actions between physicians and nurses. Nurses are responsible for the treatment of the patient and for capturing the wound documentation. The physicians are responsible for deciding and prescribing the correct wound treatment. These responsibilities inclusively define a communication process that has been reflected in this work. This work proposes how to apply the process to a telemedical scenario for homecare treatment that may involve more than one institution.

Monetary aspects concerning billing procedures have not been considered in this work.

## **5.2 Telemedical wound care**

Litzinger et al describe in (119) a portable tele consulting device which is developed for regularly scheduled nursing visits of patients with stage 3, stage 4 pressure and/or infected wounds. The patients have been visited at home and attended together with the visiting nurse a tele-consulting session. During this video session a consultant made the treatment suggestions. The consultant was responsible to document the treatment suggestions, while the nurse at the patient's home stored the digital images of the wounds on a memory card. The nurse at the patient's home also provided the wound care after listening to the consultant's suggestions.

Litzinger et al. describe a telemedicine scenario for home care patients that differs from the scenario described in this work as follows:

- The telemedicine scenario in (119) is based on a mobile video conferencing device. My work describes a distributed information system exchanging structured patient information for communicating and documenting patient information. This means that the documentation is an integrated part of the communication and has not to be done separately like in (119).
- Litzinger et al. do not mention different institutions involved in the wound management scenario. Litzinger et al do not mention if the system they used is proprietary or based on open standards in a way that other institutions may implement their own video conferencing units.
- A comprehensive wound management process is not defined exclusively. The actors in the video conferencing session are not guided through a wound management process.
- My work does originally not promote a life video session where all actors including the patient see each other and talk with each other. Nevertheless because of the standards used in this work a video conferencing tool



could be enhanced by a digital data capturing unit documenting and communicating patient related information.

In (120) Wilbright et al. describe a very similar telemedicine scenario as Litzinger but for treating patients with diabetic foot ulceration. The nurse at the patient's home was equipped with an AMD 2500 handheld camera that transmitted the real time, close-up images to the consultants at one dedicated site. Wilbright et al did neither mention a wound management process nor did they capture structured patient information.

The authors of (121-124) examined the feasibility of telemedical wound care by transmitting wound images captured by a nurse at the patient's site to a consultant at one dedicated site. The authors of these publications (121-124) conclude that telemedical wound care is feasible, improves the situation for the patient and that the patient's acceptance is high in a predominant number of cases. In (122) the patient's wound images and selected clinical data have been transmitted via a web application especially made for teledermatological monitoring of leg ulcers. The authors of (122), however, neither mentioned any communication standards nor agreed to a transparent communication process (wound management process) to assure the quality of the treatment.

### **5.3 Ontologies and ISO 13606**

There are several benefits using OWL to represent ISO 13606 archetypes. While the consistency of knowledge cannot be guaranteed by using an ADL-based archetype construction it can be granted by using an OWL-based representation. The problem is that the archetype model itself has no information about the reference model because these models are separately used. Hence instances of the archetype model (archetypes) do not refer to the reference model consistently

and in a closed environment (125). Another advantage of using OWL is the larger research community working on the development of OWL. While OWL 1.0 was released in 2004, OWL 2.0 is already available (125).

Different mechanisms are proposed for transforming ADL to OWL. For example, in (126) , they modeled the openEHR archetypes in OWL, creating syntactic compliance but without the semantic interpretation. OpenEHR is an architecture similar the ISO 13606 standard. The openEHR architecture facilitates communicating and representing EHRs, that also follows the dual model architecture approach. OpenEHR uses ADL for expressing archetypes as well, however openEHR is not considered being a standard like ISO 13606 (127). In another project (128), they mapped ADL-archetypes into OWL, but just translated the ADL expressions into OWL, again, without the semantic interpretation.

The authors of (125) describe a promising way to transform ADL into OWL without compromising the semantic interpretation of the Archetype Model. This mechanism of transforming ADL into OWL is also implemented in the LinkEHR (129) tool, which is a tool used for modeling ISO 13606 or openEHR archetypes in ADL.

As the ISO 13606 clinical standard is based on the dual model-based architecture, the main purpose is to have both, the reference model and the archetype model, represented in OWL. For example, the Archetype Model comprises concepts like *archetype*, *archetype description* or *archetype term*, whereas *folder*, *composition* or *section* are classes of the Reference Model. The results of (130) can be summarized as two main ontologies:

- EN13606-SP: representing the clinical data types and structures from the reference model.

- EN13606-AR: representing the archetype model ontology and includes the archetype model classes. Furthermore it imports the EN13606-SP ontology.

Because of the combination of these ontologies, they facilitate a more natural representation of the domain concepts of archetypes than using ADL to formulate the same domain concept. Thus it is possible to access all information concerning the same clinical term (125, 131).

In this work I decided to use and advance the approach of the combined ontologies EN13606-SP and EN13606-AR of (130). I extended the ontologies to meet the criteria of ISO13606 part 1 which means to additionally store patient related information as individuals of the class EHR\_EXTRACT. This allows the exchange of patient related information as ontology structured individuals. The individuals comprise the relations and the individuals of their archetypes. Queries on individuals e.g. using archetypes can be easily performed using ontology reasoning engines and without implementing complicated database or object frameworks. Another advantage is the data structure of the ontology directly reflects the ISO 13606 reference model which is quite similar to XML-schemata.

#### **5.4 Ontologies and clinical guidelines**

The authors of (13) address an interoperability problem concerning clinical guideline interchange formats. The authors mention 2 major problems that come along with medical guideline interchange formats and execution engines:

- “the failure of integration of guideline implementations with clinical workflows” (15, 16) and

- “the complexity of fully integrated decision support systems due to the nature of heterogeneous set of clinical applications needed to be involved in the decision process” (132).

The authors of (13) describe the mapping of GLIF to an ontology representation. This ontology representation extends GLIF and facilitates the integration of GLIF with EHR standards. Furthermore they establish an OWL-S model for semantic web service discovery and to mediate between web-service communication (and execution) and their internal GLIF ontology that access the patient data.

The work mentioned in (13) differs from the approach described in this work in various facets:

- In (13) a clinical guideline is not executed on distributed sites. The clinical guideline is executed at a certain site but retrieves the patient information from distributed hospital information systems. In my work the process description (of the wound management process) serves to describe a communication between institutions that work together and exchange patient related information. The process identifies the communicating parties and the patient related information to perform certain actions or a treatment together in a distributed environment.
- In (13) OWL and OWL-S are used to describe web services which are provided by different information systems. Mappings have to be done manually (also in OWL), once for every system to map the patient related information provided through the web services to the “GLIF RIM ontology”. The GLIF RIM ontology is accessed by the guideline execution engine. In my work every communication system must be at least aware of the communication process provided in OWL-S. Ideally every system should

implement an OWL-S reasoning engine which automatically interprets the communication process.

- The work described in (13) focuses on HL7 Reference Information Model, whereas I am focusing on the ISO 13606 communication standard for exchanging patient related information.

The authors of (13) argue that ontologies have been successfully used for representing and supporting implementations of clinical guidelines by various efforts (133, 134).

In (134) the authors describe an approach of implementing clinical guidelines with DAML/OIL. DAML/OIL is the direct ancestor of OWL and OWL-S. They utilized DAML/OIL for defining a simplified Unified Medical Language System (UMLS) ontology to structure patient related information. In my work I am using the ISO 13606 information structure to access patient related information. Furthermore the authors of (134) used DAML/OIL to define a of a laboratory procedures ontology for describing clinical practice guidelines as “context-based task ontologies (CTO)” that can be automatically interpreted. In my work I am using the OWL-S control constructs to define the wound management process.

In (135) the authors describe an approach of clinical practice guidelines using OWL-S. They chose OWL-S because of its ability to semantically describe web-services. For modeling the control flow within the clinical practice guideline they use the process model of OWL-S. The authors of (135) describe a data set ontology that provides the patient related information. This ontology is accessed by a web service. In my work the patient related information are provided by web services that access the ISO 13606 compliant information repository. Furthermore Casteleiro et al describes in (135) two more services: the guideline clinical information service which is responsible for accessing the relevant clinical prac-

tice guideline and a guideline recommendation service, which evaluates the patient condition. In my approach the execution of extended OWL-S ontology evaluates the next step of the wound management process. The information needed for the evaluation is either provided by the user interface or accessed from the EHR\_EXTRACT.

More about OWL-S and the process model can be read in the section 2.5.

## 5.5 Interoperability in health care data exchange

This section gives you an overview of the state of the art of the different levels of interoperability. This section also gives an introduction about the problems and challenges in this area.

Based on the SemanticHEALTH project (136, 137), the definitions for interoperability and semantic interoperability (SIOp) in health care are:

Health system **interoperability** is the ability, facilitated by ICT applications and systems,

- to exchange, understand and act on citizens/patients and other health-related information and knowledge
- among linguistically and culturally disparate health professionals, patients and other actors and organizations
- within and across health system jurisdictions in a collaborative manner.

Furthermore **semantic interoperability** has numerous facets:

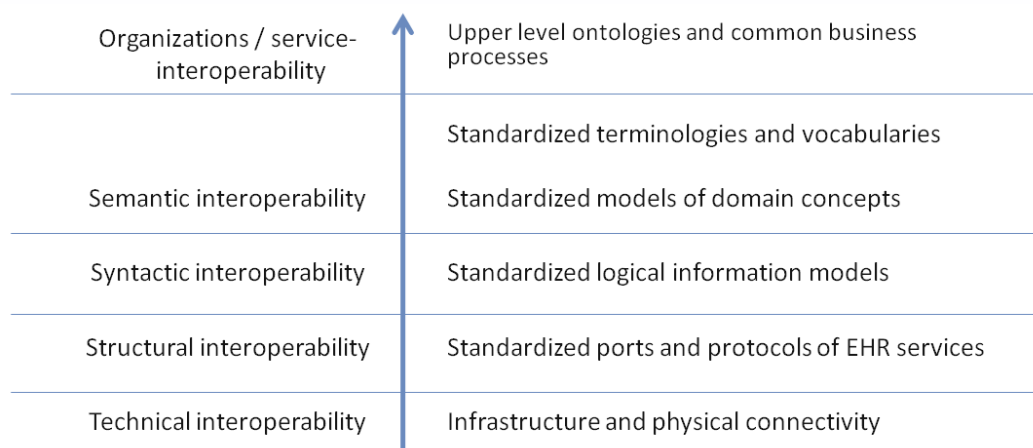
- For individual patients SIOp relevant tasks comprise assisted clinical data capture and quick access to the patient record as well as to pertinent background knowledge. It also includes quality assurance, clinical decision

support, monitoring and alerts, as well as feedback regarding quality and costs.

- For aggregated population data SIOp relevant tasks include reporting, health economics, surveillance, quality assurance, epidemiology (hypothesis formulation), bio- and tissue-banking.
- SIOp enables the meaningful linkage of research findings and knowledge to patient information, and the discovery of new knowledge from semantically coherent EHR repositories.
- In addition to precision of meaning, consistency, understandability and reproducibility are three major desiderata for semantically interoperable systems:
  - Consistency means that the receiving system must be able to recognize what has been sent, so it is a prime requirement for machine-machine communications and dictates the need for unambiguous identifiers. In other words, electronic records about patient data have to use unique identifiers for the records and the patients.
  - Understandability is essential for human communication. Humans can tolerate considerable ambiguity, but tend to focus too narrowly, so that the requirements are almost the reverse as for automated support. It is limited by the trust that the information is valid, especially with aggregated population data where the aggregation process may result in loss of information.
  - Reproducibility addresses the question of inter-individual reliability when data are collected or encoded. This holds both for individual and aggregated data.

The next higher level of semantic interoperability is service- and organizational interoperability. Service and organizational interoperability means that different

institutions work together providing different services. A process defines the relations between the services of these institutions. This means that the services have to interoperate with each other. Semantic interoperability provides the basis for service interoperability (10). Figure 41 illustrates the levels of interoperability on the left side of the arrow; on the right side of the arrow are the instances (samples) of the corresponding interoperability level. The arrow symbolizes that each level builds up upon its lower level. The lowest level is the technical interoperability; the highest level is service interoperability.



**Figure 41: Levels of interoperability derived from (10, 46)**

Now that we know what semantic interoperability means, I will describe the problems and challenges that bulge around semantic interoperability.

“For meeting the challenges of improving quality and efficiency of patient’s care including homecare and prevention, health information systems have to provide semantic interoperability supporting seamless care” (39).

This means that EHR applications have to communicate in a way that the semantic context of the captured data can be expressed and reproduced at any time and in any health care system.



Communication- and electronic interpretation- requirements for a semantically interoperable EHR are:

- Systems should be able to semantically share and combine health record data
- Terminology systems and medical knowledge databases should be accessed consistently
- Data quality and data consistency to enable secondary use of longitudinal and heterogeneous data for public health, research and health service management.
- Computerized protocols, alerts and care pathways should be integrated in EHR systems (138)
- EHR data should be interpreted within the right clinical context. EHR data is captured for a certain purpose and often during a certain clinical procedure; it must be guaranteed that the captured EHR data is interpreted at the recipient in the same clinical context or procedure after the communication happened.

Some of these requirements led to the development of EHR Standards. These standards however cover only the basics, necessary for representing and communicating EHR data (not the context). Three layers of artifacts to represent meaning had already been developed and introduced to capture, represent and communicate clinical EHR data (138):

1. Generic reference models for building the base for structuring EHR data, for example ISO 13606 Part 1 (26), HL7 CDA Release 2 (139) or openEHR Reference Model (35)

2. Explicit clinical data structure definitions like openEHR archetypes (36), ISO 13606 Part 2 (27), HL7 templates (140), generic templates and data sets.
3. Clinical terminology systems like ICD (141), LOINC (142), SNOMED-CT (143).

Still service- and process interoperability as well as service support are hardly investigated related to EHR standards. I will provide a contribution to service and process interoperability, as I will describe how to apply OWL-S to web services which provide access to EHRs. Please refer to section 4.4.1.

Another problem - that still is a big challenge - is how to associate archetype structures to coding systems or terminology systems. These associations have often been seen as final step towards semantic interoperability (138), because the terminology systems should express the clinical vocabulary and meaning. In fact terminologies and coding systems have been developed to meet the requirement to express knowledge and interrelations of clinical concepts. The usage of terminologies however is still concerned with an ambiguity problem. The next sentences describe what the ambiguity problem is about:

Creating an archetype means to define the relationships of each part of the reference model (I already mentioned the reference – and archetype model in section 2.1.1 and 2.1.2) and a term (term list) to specify the meaning (vocabulary) of the structure. Modern terminology systems however try to be a universal replacement for natural language instead of a systemization of vocabulary for clinical data items. These terminologies express the vocabulary as (sometimes) large coordinated terms. Now the problem is that archetype creation and large coordinated terms have an overlapping scope. In other words there are multiple ways of expressing the same information using data structure hierarchy (archetype) or

using a single large coordinated term (138). Hence EHR systems that access semantically equivalent information expressed ambiguously can hardly recognize or interpret this information as the same.

This work uses ISO 13606 to provide semantically interoperable patient related information via web services. Furthermore this work uses the OWL-S process model to define the communication between different systems. The OWL-S process model allows describing web services semantically to enable service or institutional interoperability. Using the OWL-S process model facilitates to define a common business process that describes the data exchange between the communicating parties.

## 6 Conclusion and Perspectives

Implementing the ODAM for the clinical process model enables applications to communicate individuals and assembling them to an integrative process. Thus, several new processes for different contents could be easily implemented without reprogramming the IT infrastructure. Furthermore interchanging individuals allows transmitting information about the state during a process. This means in a real world scenario for example information about how the patient has been already treated and how he has to be further treated is exchanged. If you imagine a scenario where several attending physicians using heterogeneous IT systems have to work together and exchange information concerning one patient, process information is necessary to coordinate the communication procedure between the parties.

This approach facilitates the definition of a EBM tele wound management process for an interactive IT support for physicians and nurses.

Combining the process model with ISO 13606 in an Ontology Driven Application approach, enables decision support using all aforementioned benefits. The process builds the knowledge base for guidance and decision support, the Archetypes and EHR-extracts (patient information) trigger the workflow.

Checklists and guidelines for good clinical practice are important for high quality health care delivery. Especially in the field of integrated care applications, which are frequently well applicable for chronic disease, guided processes can improve the communication between multidisciplinary HCPs and provide knowledge about the medical state of the art. IT applications facilitating multidisciplinary, guideline-based integrated care can realize the vision of a standardized and quality assured health care. Guidelines and archetypes can be provided in repository

ries, updated and maintained by centralized institutions (WHO, health ministry, etc.). Based on a generic ontology-driven application design that allows deriving the application functionalities from archetypes and ontologies, changes or adaptations of the contents in the repositories affecting the healthcare delivery without changing the IT application, may become reality. Thus resulting in an up to date, high quality and controlled health care, pushing efficiency and making medical decision (and treatment) traceable.

This work also aims to make a contribution to the integration of legacy systems within integrated care scenarios. Especially to satisfy user acceptance of telemedical systems, it is helpful to enable a communication between legacy hospital and physician-office information systems. Developing and implementing adaptors for health information standards (like ISO13606) efficiently will play an important role. The ODAM can be a first step to understand and develop more sophisticated methods for realizing legacy system and back end adaptors.

Although we did not have difficulties creating OWL-S processes to support our medical processes, a transformation from guideline interchange formats like GLIF to OWL-S processes could allow the usage of already established electronic guidelines in a semantic web context. RubyTL (a Model Driven Development Transformation Language (144)) could be a promising tool to implement a model driven transformation from the GLIF model to the OWL-S model.

Another interesting future work could be a graphical editor for developing archetypes graphically. The LinkEHR archetype editor is complicated to use especially for clinical domain professionals without any knowledge about archetypes and the underlying reference model. It is highly recommendable that archetypes should be developed by clinical domain specialists or at least supervised by them. However, the result produced by the editor is not easily comprehensible for non

IT specialists. An editor that supports the graphical development of an archetype as an HTML input form for example could be more intuitive. Thereby the domain specialist could easily get an impression of how the archetype derived EHR components look like. However, LinkEHR was sufficient for our work.

Another feature of a graphical archetype editor could be a graphical process designer for defining clinical process definitions. Although we already used a plugin for Protégé to graphically develop OWL-S processes, optimizations for non IT professionals could be done to embed 13606 archetypes more intuitively into the OWL-S processes.

## **7 Acknowledgements**

Prof. Binder, my PhD supervisor, his assistant Jessika Weingast and the Telemedicine group of Prof. Binder gave me the necessary medical inspiration, know-how and infrastructure that made this work possible.

Prof. Duftschmid, who has been my technical reviewer, assured the quality of this work. He prevented me from academic pitfalls.

My special thanks go also to Markus Dorn and Hilal Tekoglu. Under my supervision Markus Dorn developed some major parts of the prototype system for his master thesis.

Hilal Tekoglu reviewed this work.

## 8 Appendix

### 8.1 Archetypes in ADL format

#### 8.1.1 Wound management general patient info archetype

```
archetype (adl_version=1.4)
  CEN-EN13606-COMPOSITION.WMGeneralPatientInfo.v1

concept
  [at0000]

language
  original_language = <[ISO_639-1::en-us]>

description
  original_author = <
    ["date"] = <"20110505">
  >
  lifecycle_state = <"Draft">
  details = <
    ["en-us"] = <
      language = <[ISO_639-1::en-us]>
    >
  >
  >

definition
  COMPOSITION[at0000] occurrences matches {1..1} matches { -- WMGeneralPatientInfo
    content existence matches {0..1} cardinality matches {0..*; unordered} matches {
      ENTRY[at0001] occurrences matches {0..*} matches { -- General Patient Info
        items existence matches {0..1} cardinality matches {0..*; unordered; unique}
      matches {
        ELEMENT[at0002] occurrences matches {0..*} matches { -- Weight
          value existence matches {0..1} matches {
            SIMPLE_TEXT[at0003] occurrences matches {0..1} matches { --
SIMPLE_TEXT
              originalText existence matches {0..1} matches {././}
            }
          }
        }
        ELEMENT[at0004] occurrences matches {0..*} matches { -- Height
          value existence matches {0..1} matches {
            SIMPLE_TEXT[at0005] occurrences matches {0..1} matches { --
SIMPLE_TEXT
              originalText existence matches {0..1} matches {././}
            }
          }
        }
      }
    }
    ELEMENT[at0006] occurrences matches {0..*} matches { -- Occlusive Artery
      Disease
      value existence matches {0..1} matches {
        SIMPLE_TEXT[at0034] occurrences matches {0..1} matches {*} -- not
available
        SIMPLE_TEXT[at0035] occurrences matches {0..1} matches {*} -- yes
        SIMPLE_TEXT[at0036] occurrences matches {0..1} matches {*} -- no
        SIMPLE_TEXT[at0037] occurrences matches {0..1} matches {*} -- I
(Fontaine)
        SIMPLE_TEXT[at0038] occurrences matches {0..1} matches {*} -- II
(Fontaine)
        SIMPLE_TEXT[at0039] occurrences matches {0..1} matches {*} -- III
(Fontaine)
        SIMPLE_TEXT[at0040] occurrences matches {0..1} matches {*} -- IV
(Fontaine)
      }
    }
    ELEMENT[at0008] occurrences matches {0..*} matches { -- Diabetes Mellitus
      value existence matches {0..1} matches {
        SIMPLE_TEXT[at0034] occurrences matches {0..1} matches {*} -- not
available
        SIMPLE_TEXT[at0035] occurrences matches {0..1} matches {*} -- yes
        SIMPLE_TEXT[at0036] occurrences matches {0..1} matches {*} -- no
        SIMPLE_TEXT[at0044] occurrences matches {0..1} matches {*} --
type I
        SIMPLE_TEXT[at0045] occurrences matches {0..1} matches {*} --
type II
      }
    }
  }
  ELEMENT[at0010] occurrences matches {0..*} matches { -- Pressure
```



```

value existence matches {0..1} matches {
available      SIMPLE_TEXT[at0034] occurrences matches {0..1} matches {*} -- not
              SIMPLE_TEXT[at0035] occurrences matches {0..1} matches {*} -- yes
              SIMPLE_TEXT[at0036] occurrences matches {0..1} matches {*} -- no
            }
}
ELEMENT[at0012] occurrences matches {0..*} matches { -- Trauma
value existence matches {0..1} matches {
available      SIMPLE_TEXT[at0034] occurrences matches {0..1} matches {*} -- not
              SIMPLE_TEXT[at0035] occurrences matches {0..1} matches {*} -- yes
              SIMPLE_TEXT[at0036] occurrences matches {0..1} matches {*} -- no
            }
}
ELEMENT[at0014] occurrences matches {0..*} matches { -- CVI
value existence matches {0..1} matches {
available      SIMPLE_TEXT[at0034] occurrences matches {0..1} matches {*} -- not
              SIMPLE_TEXT[at0035] occurrences matches {0..1} matches {*} -- yes
              SIMPLE_TEXT[at0036] occurrences matches {0..1} matches {*} -- no
              SIMPLE_TEXT[at0041] occurrences matches {0..1} matches {*} -- I
(Widmer)      SIMPLE_TEXT[at0042] occurrences matches {0..1} matches {*} -- II
(Widmer)      SIMPLE_TEXT[at0043] occurrences matches {0..1} matches {*} -- III
(Widmer)
            }
}
ELEMENT[at0016] occurrences matches {0..*} matches { -- Immobility
value existence matches {0..1} matches {
available      SIMPLE_TEXT[at0034] occurrences matches {0..1} matches {*} -- not
              SIMPLE_TEXT[at0035] occurrences matches {0..1} matches {*} -- yes
              SIMPLE_TEXT[at0036] occurrences matches {0..1} matches {*} -- no
            }
}
ELEMENT[at0018] occurrences matches {0..*} matches { -- Nicotine Abuse
value existence matches {0..1} matches {
available      SIMPLE_TEXT[at0034] occurrences matches {0..1} matches {*} -- not
              SIMPLE_TEXT[at0035] occurrences matches {0..1} matches {*} -- yes
              SIMPLE_TEXT[at0036] occurrences matches {0..1} matches {*} -- no
            }
}
ELEMENT[at0020] occurrences matches {0..*} matches { -- Malnutrition
value existence matches {0..1} matches {
available      SIMPLE_TEXT[at0034] occurrences matches {0..1} matches {*} -- not
              SIMPLE_TEXT[at0035] occurrences matches {0..1} matches {*} -- yes
              SIMPLE_TEXT[at0036] occurrences matches {0..1} matches {*} -- no
            }
}
ELEMENT[at0022] occurrences matches {0..*} matches { -- Reduced General
Condition
value existence matches {0..1} matches {
available      SIMPLE_TEXT[at0034] occurrences matches {0..1} matches {*} -- not
              SIMPLE_TEXT[at0035] occurrences matches {0..1} matches {*} -- yes
              SIMPLE_TEXT[at0036] occurrences matches {0..1} matches {*} -- no
            }
}
ELEMENT[at0024] occurrences matches {0..*} matches { -- Metabolic Disor-
der
value existence matches {0..1} matches {
available      SIMPLE_TEXT[at0034] occurrences matches {0..1} matches {*} -- not
              SIMPLE_TEXT[at0035] occurrences matches {0..1} matches {*} -- yes
              SIMPLE_TEXT[at0036] occurrences matches {0..1} matches {*} -- no
            }
}
ELEMENT[at0026] occurrences matches {0..*} matches { -- Cortisone Taking
value existence matches {0..1} matches {
available      SIMPLE_TEXT[at0034] occurrences matches {0..1} matches {*} -- not
              SIMPLE_TEXT[at0035] occurrences matches {0..1} matches {*} -- yes
              SIMPLE_TEXT[at0036] occurrences matches {0..1} matches {*} -- no
            }
}
ELEMENT[at0028] occurrences matches {0..*} matches { -- Immunosuppression
value existence matches {0..1} matches {
available      SIMPLE_TEXT[at0034] occurrences matches {0..1} matches {*} -- not
              SIMPLE_TEXT[at0035] occurrences matches {0..1} matches {*} -- yes
              SIMPLE_TEXT[at0036] occurrences matches {0..1} matches {*} -- no
            }
}
ELEMENT[at0030] occurrences matches {0..*} matches { -- Infectious Dis-
eases

```

```

        value existence matches {0..1} matches {
SIMPLE_TEXT          SIMPLE_TEXT[at0031] occurrences matches {0..1} matches { --
                    }
                    originalText existence matches {0..1} matches {/./}
                    }
        }
    ELEMENT[at0032] occurrences matches {0..*} matches { -- Comments
        value existence matches {0..1} matches {
SIMPLE_TEXT          SIMPLE_TEXT[at0033] occurrences matches {0..1} matches { --
                    }
                    originalText existence matches {0..1} matches {/./}
                    }
        }
    }
}

ontology
terminologies_available = <...>
term_definitions = <
  ["en-us"] = <
    items = <
      ["at0000"] = <
        text = <"WMGeneralPatientInfo">
        description = <"WMGeneralPatientInfo">
      >
      ["at0001"] = <
        text = <"General Patient Info">
        description = <"This is a ENTRY object">
      >
      ["at0002"] = <
        text = <"Weight">
        description = <"This is a ELEMENT object">
      >
      ["at0003"] = <
        text = <"SIMPLE TEXT">
        description = <"This is a SIMPLE_TEXT object">
      >
      ["at0004"] = <
        text = <"Height">
        description = <"This is a ELEMENT object">
      >
      ["at0005"] = <
        text = <"SIMPLE TEXT">
        description = <"This is a SIMPLE_TEXT object">
      >
      ["at0006"] = <
        text = <"Occlusive Artery Disease">
        description = <"This is a ELEMENT object">
      >
      ["at0008"] = <
        text = <"Diabetes Mellitus">
        description = <"This is a ELEMENT object">
      >
      ["at0010"] = <
        text = <"Pressure">
        description = <"This is a ELEMENT object">
      >
      ["at0011"] = <
        text = <"SIMPLE TEXT">
        description = <"This is a SIMPLE_TEXT object">
      >
      ["at0012"] = <
        text = <"Trauma">
        description = <"This is a ELEMENT object">
      >
      ["at0013"] = <
        text = <"SIMPLE TEXT">
        description = <"This is a SIMPLE_TEXT object">
      >
      ["at0014"] = <
        text = <"CVI">
        description = <"This is a ELEMENT object">
      >
      ["at0016"] = <
        text = <"Immobility">
        description = <"This is a ELEMENT object">
      >
      ["at0017"] = <
        text = <"SIMPLE TEXT">
        description = <"This is a SIMPLE_TEXT object">
      >
      ["at0018"] = <
        text = <"Nicotine Abuse">
        description = <"This is a ELEMENT object">
      >
    >
  >

```

```

["at0019"] = <
  text = <"SIMPLE TEXT">
  description = <"This is a SIMPLE_TEXT object">
>
["at0020"] = <
  text = <"Malnutrition">
  description = <"This is a ELEMENT object">
>
["at0021"] = <
  text = <"SIMPLE TEXT">
  description = <"This is a SIMPLE_TEXT object">
>
["at0022"] = <
  text = <"Reduced General Condition">
  description = <"This is a ELEMENT object">
>
["at0023"] = <
  text = <"SIMPLE TEXT">
  description = <"This is a SIMPLE_TEXT object">
>
["at0024"] = <
  text = <"Metabolic Disorder">
  description = <"This is a ELEMENT object">
>
["at0025"] = <
  text = <"SIMPLE TEXT">
  description = <"This is a SIMPLE_TEXT object">
>
["at0026"] = <
  text = <"Cortisone Taking">
  description = <"This is a ELEMENT object">
>
["at0027"] = <
  text = <"SIMPLE TEXT">
  description = <"This is a SIMPLE_TEXT object">
>
["at0028"] = <
  text = <"Immunosuppression">
  description = <"This is a ELEMENT object">
>
["at0029"] = <
  text = <"SIMPLE TEXT">
  description = <"This is a SIMPLE_TEXT object">
>
["at0030"] = <
  text = <"Infectious Diseases">
  description = <"This is a ELEMENT object">
>
["at0031"] = <
  text = <"SIMPLE TEXT">
  description = <"This is a SIMPLE_TEXT object">
>
["at0032"] = <
  text = <"Comments">
  description = <"This is a ELEMENT object">
>
["at0033"] = <
  text = <"SIMPLE TEXT">
  description = <"This is a SIMPLE_TEXT object">
>
["at0034"] = <
  text = <"not available">
  description = <"This is a SIMPLE_TEXT object">
>
["at0035"] = <
  text = <"yes">
  description = <"This is a SIMPLE_TEXT object">
>
["at0036"] = <
  text = <"no">
  description = <"This is a SIMPLE_TEXT object">
>
["at0037"] = <
  text = <"I (Fontaine)">
  description = <"This is a SIMPLE_TEXT object">
>
["at0038"] = <
  text = <"II (Fontaine)">
  description = <"This is a SIMPLE_TEXT object">
>
["at0039"] = <
  text = <"III (Fontaine)">
  description = <"This is a SIMPLE_TEXT object">
>
["at0040"] = <
  text = <"IV (Fontaine)">
  description = <"This is a SIMPLE_TEXT object">
>
["at0041"] = <

```

```

        text = <"I (Widmer)">
        description = <"This is a SIMPLE_TEXT object">
    >
["at0042"] = <
    text = <"II (Widmer)">
    description = <"This is a SIMPLE_TEXT object">
>
["at0043"] = <
    text = <"III (Widmer)">
    description = <"This is a SIMPLE_TEXT object">
>
["at0044"] = <
    text = <"type I">
    description = <"This is a SIMPLE_TEXT object">
>
["at0045"] = <
    text = <"type II">
    description = <"This is a SIMPLE_TEXT object">
>
    >
>
>
>
constraint_definitions = <
>
term_binding = <
>
constraint_binding = <
>

```

## 8.1.2 Wound management archetype – wound specific

```

archetype (adl version=1.4)
  CEN-EN13606-COMPOSITION.WMWoundSpecific.v1

concept
  [at0000]

language
  original_language = <[ISO_639-1::en-us]>

description
  original_author = <
    ["date"] = <"20110505">
  >
  lifecycle_state = <"Draft">
  details = <
    ["en-us"] = <
      language = <[ISO_639-1::en-us]>
    >
  >
>

definition
  COMPOSITION[at0000] occurrences matches {1..1} matches { -- WMWoundSpecific
    content existence matches {0..1} cardinality matches {0..*; unordered} matches {
      ENTRY[at0001] occurrences matches {0..*} matches { -- Wound Specific Info
        items existence matches {0..1} cardinality matches {0..*; unordered; unique}
      }
    }
  matches {
    ELEMENT[at0002] occurrences matches {0..*} matches { -- Localization
      value existence matches {0..1} matches {
        SIMPLE_TEXT[at0003] occurrences matches {0..1} matches { --
          originalText existence matches {0..1} matches {/.*/}
        }
      }
    }
    ELEMENT[at0004] occurrences matches {0..*} matches { -- Wound
      value existence matches {0..1} matches {
        SIMPLE_TEXT[at0087] occurrences matches {0..1} matches {*} --
        SIMPLE_TEXT[at0088] occurrences matches {0..1} matches {*} --
        SIMPLE_TEXT[at0089] occurrences matches {0..1} matches {*} --
        SIMPLE_TEXT[at0090] occurrences matches {0..1} matches {*} --
        SIMPLE_TEXT[at0091] occurrences matches {0..1} matches {*} --
        SIMPLE_TEXT[at0092] occurrences matches {0..1} matches {*} --
        SIMPLE_TEXT[at0093] occurrences matches {0..1} matches {*} --
        SIMPLE_TEXT[at0094] occurrences matches {0..1} matches {*} --
        SIMPLE_TEXT[at0095] occurrences matches {0..1} matches {*} --
      }
    }
  }

ulcus cruris
ulcus cruris arteriosum
ulcus cruris venosum
ulcus cruris mixtum
decubitus
decubitus grade I
decubitus grade II
decubitus grade III
decubitus grade IV

```

```

diabetic foot syndrome      SIMPLE_TEXT[at0096] occurrences matches {0..1} matches {*} --
postoperative impaired wound healing
                             SIMPLE_TEXT[at0097] occurrences matches {0..1} matches {*} --
                             }
                             }
ELEMENT[at0006] occurrences matches {0..*} matches { -- Duration
value existence matches {0..1} matches {
SIMPLE_TEXT[at0007] occurrences matches {0..1} matches { --
SIMPLE_TEXT
                             originalText existence matches {0..1} matches {/./}
                             }
                             }
ELEMENT[at0008] occurrences matches {0..*} matches { -- Recurrence count
value existence matches {0..1} matches {
SIMPLE_TEXT[at0009] occurrences matches {0..1} matches { --
SIMPLE_TEXT
                             originalText existence matches {0..1} matches {/./}
                             }
                             }
ELEMENT[at0010] occurrences matches {0..*} matches { -- Length
value existence matches {0..1} matches {
SIMPLE_TEXT[at0011] occurrences matches {0..1} matches { --
SIMPLE_TEXT
                             originalText existence matches {0..1} matches {/./}
                             }
                             }
ELEMENT[at0012] occurrences matches {0..*} matches { -- Width
value existence matches {0..1} matches {
SIMPLE_TEXT[at0013] occurrences matches {0..1} matches { --
originalText existence matches {0..1} matches {/./}
}
}
ELEMENT[at0014] occurrences matches {0..*} matches { -- Depth
value existence matches {0..1} matches {
SIMPLE_TEXT[at0015] occurrences matches {0..1} matches { --
SIMPLE_TEXT
                             originalText existence matches {0..1} matches {/./}
                             }
                             }
}
CLUSTER[at0018] occurrences matches {0..*} matches { -- Reason
parts existence matches {0..1} cardinality matches {0..*}; unordered;
unique) matches {
    ing
    ELEMENT[at0019] occurrences matches {0..*} matches { -- granulat-
        value existence matches {0..1} matches {
        SIMPLE_TEXT[at0020] occurrences matches {0..1} matches {*}
        }
    }
    ELEMENT[at0021] occurrences matches {0..*} matches { -- epitheli-
        value existence matches {0..1} matches {
        SIMPLE_TEXT[at0022] occurrences matches {0..1} matches {*}
        }
    }
    ELEMENT[at0023] occurrences matches {0..*} matches { -- firbrin
        value existence matches {0..1} matches {
        SIMPLE_TEXT[at0024] occurrences matches {0..1} matches {*}
        }
    }
    ELEMENT[at0025] occurrences matches {0..*} matches { -- necrotic
        value existence matches {0..1} matches {
        SIMPLE_TEXT[at0026] occurrences matches {0..1} matches {*}
        }
    }
}
}
CLUSTER[at0027] occurrences matches {0..*} matches { -- Border
parts existence matches {0..1} cardinality matches {0..*}; unordered;
unique) matches {
    ELEMENT[at0028] occurrences matches {0..*} matches { -- bland
        value existence matches {0..1} matches {
        SIMPLE_TEXT[at0029] occurrences matches {0..1} matches {*}
        }
    }
    ELEMENT[at0030] occurrences matches {0..*} matches { -- reddened
        value existence matches {0..1} matches {
        SIMPLE_TEXT[at0031] occurrences matches {0..1} matches {*}
        }
    }
}
-- reddened

```

```

    }
  }
  ELEMENT[at0016] occurrences matches {0..*} matches { -- edematous
    value existence matches {0..1} matches {
      SIMPLE_TEXT[at0017] occurrences matches {0..1} matches {*}
    }
  }
  -- edematous
  }
  ELEMENT[at0032] occurrences matches {0..*} matches { -- macer-
    value existence matches {0..1} matches {
      SIMPLE_TEXT[at0033] occurrences matches {0..1} matches {*}
    }
  }
  -- maceratus
  }
  ELEMENT[at0034] occurrences matches {0..*} matches { -- epitheli-
    value existence matches {0..1} matches {
      SIMPLE_TEXT[at0035] occurrences matches {0..1} matches {*}
    }
  }
  -- epithelializing
  }
  }
  CLUSTER[at0036] occurrences matches {0..*} matches { -- Surrounding
    parts existence matches {0..1} cardinality matches {0..*}; unordered;
  }
  unique) matches {
    ELEMENT[at0037] occurrences matches {0..*} matches { -- bland
      value existence matches {0..1} matches {
        SIMPLE_TEXT[at0038] occurrences matches {0..1} matches {*}
      }
    }
    -- bland
    }
    ELEMENT[at0039] occurrences matches {0..*} matches { -- reddened
      value existence matches {0..1} matches {
        SIMPLE_TEXT[at0040] occurrences matches {0..1} matches {*}
      }
    }
    -- reddened
    }
    ELEMENT[at0041] occurrences matches {0..*} matches { -- dry
      value existence matches {0..1} matches {
        SIMPLE_TEXT[at0042] occurrences matches {0..1} matches {*}
      }
    }
    -- dry
    }
    ELEMENT[at0043] occurrences matches {0..*} matches { -- edematous
      value existence matches {0..1} matches {
        SIMPLE_TEXT[at0044] occurrences matches {0..1} matches {*}
      }
    }
    -- edematous
    }
    ELEMENT[at0045] occurrences matches {0..*} matches { -- macer-
      value existence matches {0..1} matches {
        SIMPLE_TEXT[at0046] occurrences matches {0..1} matches {*}
      }
    }
    -- maceratus
    }
    ELEMENT[at0047] occurrences matches {0..*} matches { -- livid
      value existence matches {0..1} matches {
        SIMPLE_TEXT[at0048] occurrences matches {0..1} matches {*}
      }
    }
    -- livid
    }
    ELEMENT[at0049] occurrences matches {0..*} matches { -- hyper-
      value existence matches {0..1} matches {
        SIMPLE_TEXT[at0050] occurrences matches {0..1} matches {*}
      }
    }
    -- hyperthermic
    }
  }
  }
  ELEMENT[at0051] occurrences matches {0..*} matches { -- Healing Stage
    value existence matches {0..1} matches {
      SIMPLE_TEXT[at0052] occurrences matches {0..1} matches {*} --
      SIMPLE_TEXT[at0053] occurrences matches {0..1} matches {*} --
      SIMPLE_TEXT[at0054] occurrences matches {0..1} matches {*} --
    }
  }
  CLUSTER[at0055] occurrences matches {0..*} matches { -- Exudate
    parts existence matches {0..1} cardinality matches {0..*}; unordered;
  }
  unique) matches {
    ELEMENT[at0056] occurrences matches {0..*} matches { -- serous
      value existence matches {0..1} matches {

```

```

        SIMPLE_TEXT[at0057] occurrences matches {0..1} matches {*}
    }
}
ELEMENT[at0058] occurrences matches {0..*} matches { -- sanguineous
    value existence matches {0..1} matches {
        SIMPLE_TEXT[at0059] occurrences matches {0..1} matches {*}
    }
}
ELEMENT[at0060] occurrences matches {0..*} matches { -- purulent
    value existence matches {0..1} matches {
        SIMPLE_TEXT[at0061] occurrences matches {0..1} matches {*}
    }
}
ELEMENT[at0062] occurrences matches {0..*} matches { -- greenish
    value existence matches {0..1} matches {
        SIMPLE_TEXT[at0063] occurrences matches {0..1} matches {*}
    }
}
ELEMENT[at0064] occurrences matches {0..*} matches { -- ELEMENT
    value existence matches {0..1} matches {
        SIMPLE_TEXT[at0065] occurrences matches {0..1} matches {*}
        SIMPLE_TEXT[at0066] occurrences matches {0..1} matches {*}
        SIMPLE_TEXT[at0067] occurrences matches {0..1} matches {*}
        SIMPLE_TEXT[at0068] occurrences matches {0..1} matches {*}
    }
}
}
ELEMENT[at0069] occurrences matches {0..*} matches { -- Woundsmell
    value existence matches {0..1} matches {
        SIMPLE_TEXT[at0070] occurrences matches {0..1} matches {*} -- not available
        SIMPLE_TEXT[at0071] occurrences matches {0..1} matches {*} -- yes
        SIMPLE_TEXT[at0072] occurrences matches {0..1} matches {*} -- no
    }
}
ELEMENT[at0073] occurrences matches {0..*} matches { -- Pain
    value existence matches {0..1} matches {
        SIMPLE_TEXT[at0070] occurrences matches {0..1} matches {*} -- not available
        SIMPLE_TEXT[at0072] occurrences matches {0..1} matches {*} -- no
        SIMPLE_TEXT[at0074] occurrences matches {0..1} matches {*} -- yes
        SIMPLE_TEXT[at0075] occurrences matches {0..1} matches {*} -- yes
        SIMPLE_TEXT[at0076] occurrences matches {0..1} matches {*} -- yes
        SIMPLE_TEXT[at0077] occurrences matches {0..1} matches {*} -- yes
        SIMPLE_TEXT[at0078] occurrences matches {0..1} matches {*} -- yes
        SIMPLE_TEXT[at0079] occurrences matches {0..1} matches {*} -- yes
        SIMPLE_TEXT[at0080] occurrences matches {0..1} matches {*} -- yes
        SIMPLE_TEXT[at0081] occurrences matches {0..1} matches {*} -- yes
        SIMPLE_TEXT[at0082] occurrences matches {0..1} matches {*} -- yes
        SIMPLE_TEXT[at0083] occurrences matches {0..1} matches {*} -- yes
        SIMPLE_TEXT[at0084] occurrences matches {0..1} matches {*} -- yes
    }
}
ELEMENT[at0085] occurrences matches {0..*} matches { -- Comments
    value existence matches {0..1} matches {
        SIMPLE_TEXT[at0086] occurrences matches {0..1} matches {*} -- SIMPLE_TEXT
    }
}
ELEMENT[at0098] occurrences matches {0..*} matches { -- Wound Images
    value existence matches {0..1} matches {
        ED[at0099] occurrences matches {0..1} matches { -- ED
            data existence matches {0..1} matches {*}
            mediaType existence matches {0..1} matches {
                CS[at0108] occurrences matches {0..1} matches { --
                    codeValue existence matches {0..1} matches {"jpeg"}
                    codingSchemeName existence matches {0..1} matches
{"IANA::MIME_TYPES"}
            }
        }
    }
}

```





```

description = <"This is a ELEMENT object">
>
["at0015"] = <
  text = <"SIMPLE_TEXT">
  description = <"This is a SIMPLE_TEXT object">
>
["at0018"] = <
  text = <"Reason">
  description = <"This is a CLUSTER object">
>
["at0019"] = <
  text = <"granulating">
  description = <"This is a ELEMENT object">
>
["at0020"] = <
  text = <"granulating">
  description = <"This is a SIMPLE_TEXT object">
>
["at0021"] = <
  text = <"epithelializing">
  description = <"This is a ELEMENT object">
>
["at0022"] = <
  text = <"epithelializing">
  description = <"This is a SIMPLE_TEXT object">
>
["at0023"] = <
  text = <"firbrin">
  description = <"This is a ELEMENT object">
>
["at0024"] = <
  text = <"fibrin">
  description = <"This is a SIMPLE_TEXT object">
>
["at0025"] = <
  text = <"necrotic">
  description = <"This is a ELEMENT object">
>
["at0026"] = <
  text = <"necrotic">
  description = <"This is a SIMPLE_TEXT object">
>
["at0027"] = <
  text = <"Border">
  description = <"This is a CLUSTER object">
>
["at0028"] = <
  text = <"bland">
  description = <"This is a ELEMENT object">
>
["at0029"] = <
  text = <"bland">
  description = <"This is a SIMPLE_TEXT object">
>
["at0030"] = <
  text = <"reddened">
  description = <"This is a ELEMENT object">
>
["at0031"] = <
  text = <"reddened">
  description = <"This is a SIMPLE_TEXT object">
>
["at0016"] = <
  text = <"edematous">
  description = <"This is a ELEMENT object">
>
["at0017"] = <
  text = <"edematous">
  description = <"This is a SIMPLE_TEXT object">
>
["at0032"] = <
  text = <"maceratous">
  description = <"This is a ELEMENT object">
>
["at0033"] = <
  text = <"maceratous">
  description = <"This is a SIMPLE_TEXT object">
>
["at0034"] = <
  text = <"epithelializing">
  description = <"This is a ELEMENT object">
>
["at0035"] = <
  text = <"epithelializing">
  description = <"This is a SIMPLE_TEXT object">
>
["at0036"] = <
  text = <"Surrounding">
  description = <"This is a CLUSTER object">

```

```

>
["at0037"] = <
  text = <"bland">
  description = <"This is a ELEMENT object">
>
["at0038"] = <
  text = <"bland">
  description = <"This is a SIMPLE_TEXT object">
>
["at0039"] = <
  text = <"reddened">
  description = <"This is a ELEMENT object">
>
["at0040"] = <
  text = <"reddened">
  description = <"This is a SIMPLE_TEXT object">
>
["at0041"] = <
  text = <"dry">
  description = <"This is a ELEMENT object">
>
["at0042"] = <
  text = <"dry">
  description = <"This is a SIMPLE_TEXT object">
>
["at0043"] = <
  text = <"edematous">
  description = <"This is a ELEMENT object">
>
["at0044"] = <
  text = <"edematous">
  description = <"This is a SIMPLE_TEXT object">
>
["at0045"] = <
  text = <"maceratous">
  description = <"This is a ELEMENT object">
>
["at0046"] = <
  text = <"maceratous">
  description = <"This is a SIMPLE_TEXT object">
>
["at0047"] = <
  text = <"livid">
  description = <"This is a ELEMENT object">
>
["at0048"] = <
  text = <"livid">
  description = <"This is a SIMPLE_TEXT object">
>
["at0049"] = <
  text = <"hyperthermic">
  description = <"This is a ELEMENT object">
>
["at0050"] = <
  text = <"hyperthermic">
  description = <"This is a SIMPLE_TEXT object">
>
["at0051"] = <
  text = <"Healing Stage">
  description = <"This is a ELEMENT object">
>
["at0052"] = <
  text = <"cleaning">
  description = <"This is a SIMPLE_TEXT object">
>
["at0053"] = <
  text = <"granulation tissue fromation">
  description = <"This is a SIMPLE_TEXT object">
>
["at0054"] = <
  text = <"epitheliazation">
  description = <"This is a SIMPLE_TEXT object">
>
["at0055"] = <
  text = <"Exudate">
  description = <"This is a CLUSTER object">
>
["at0056"] = <
  text = <"serous">
  description = <"This is a ELEMENT object">
>
["at0057"] = <
  text = <"serous">
  description = <"This is a SIMPLE_TEXT object">
>
["at0058"] = <
  text = <"sanguineous">
  description = <"This is a ELEMENT object">
>

```

```

["at0059"] = <
  text = <"sanguineous">
  description = <"This is a SIMPLE_TEXT object">
>
["at0060"] = <
  text = <"purulent">
  description = <"This is a ELEMENT object">
>
["at0061"] = <
  text = <"purulent">
  description = <"This is a SIMPLE_TEXT object">
>
["at0062"] = <
  text = <"greenish">
  description = <"This is a ELEMENT object">
>
["at0063"] = <
  text = <"greenish">
  description = <"This is a SIMPLE_TEXT object">
>
["at0064"] = <
  text = <"ELEMENT">
  description = <"This is a ELEMENT object">
>
["at0065"] = <
  text = <"none">
  description = <"This is a SIMPLE_TEXT object">
>
["at0066"] = <
  text = <"little">
  description = <"This is a SIMPLE_TEXT object">
>
["at0067"] = <
  text = <"moderate">
  description = <"This is a SIMPLE_TEXT object">
>
["at0068"] = <
  text = <"a lot">
  description = <"This is a SIMPLE_TEXT object">
>
["at0069"] = <
  text = <"Woundsmell">
  description = <"This is a ELEMENT object">
>
["at0070"] = <
  text = <"not available">
  description = <"This is a SIMPLE_TEXT object">
>
["at0071"] = <
  text = <"yes">
  description = <"This is a SIMPLE_TEXT object">
>
["at0072"] = <
  text = <"no">
  description = <"This is a SIMPLE_TEXT object">
>
["at0073"] = <
  text = <"Pain">
  description = <"This is a ELEMENT object">
>
["at0074"] = <
  text = <"yes 0 (little)">
  description = <"This is a SIMPLE_TEXT object">
>
["at0075"] = <
  text = <"yes 1">
  description = <"This is a SIMPLE_TEXT object">
>
["at0076"] = <
  text = <"yes 2">
  description = <"This is a SIMPLE_TEXT object">
>
["at0077"] = <
  text = <"yes 3">
  description = <"This is a SIMPLE_TEXT object">
>
["at0078"] = <
  text = <"yes 4">
  description = <"This is a SIMPLE_TEXT object">
>
["at0079"] = <
  text = <"yes 5">
  description = <"This is a SIMPLE_TEXT object">
>
["at0080"] = <
  text = <"yes 6">
  description = <"This is a SIMPLE_TEXT object">
>
["at0081"] = <

```

```

        text = <"yes 7">
        description = <"This is a SIMPLE_TEXT object">
    >
    ["at0082"] = <
        text = <"yes 8">
        description = <"This is a SIMPLE_TEXT object">
    >
    ["at0083"] = <
        text = <"yes 9">
        description = <"This is a SIMPLE_TEXT object">
    >
    ["at0084"] = <
        text = <"yes 10 (extreme)">
        description = <"This is a SIMPLE_TEXT object">
    >
    ["at0085"] = <
        text = <"Comments">
        description = <"This is a ELEMENT object">
    >
    ["at0086"] = <
        text = <"SIMPLE TEXT">
        description = <"This is a SIMPLE_TEXT object">
    >
    ["at0087"] = <
        text = <"ulcus cruris">
        description = <"This is a SIMPLE_TEXT object">
    >
    ["at0088"] = <
        text = <"ulcus cruris arteriosum">
        description = <"This is a SIMPLE_TEXT object">
    >
    ["at0089"] = <
        text = <"ulcus cruris venosum">
        description = <"This is a SIMPLE_TEXT object">
    >
    ["at0090"] = <
        text = <"ulcus cruris mixtum">
        description = <"This is a SIMPLE_TEXT object">
    >
    ["at0091"] = <
        text = <"decubitus">
        description = <"This is a SIMPLE_TEXT object">
    >
    ["at0092"] = <
        text = <"decubitus grade I">
        description = <"This is a SIMPLE_TEXT object">
    >
    ["at0093"] = <
        text = <"decubitus grade II">
        description = <"This is a SIMPLE_TEXT object">
    >
    ["at0094"] = <
        text = <"decubitus grade III">
        description = <"This is a SIMPLE_TEXT object">
    >
    ["at0095"] = <
        text = <"decubitus grade IV">
        description = <"This is a SIMPLE_TEXT object">
    >
    ["at0096"] = <
        text = <"diabetic foot syndrome">
        description = <"This is a SIMPLE_TEXT object">
    >
    ["at0097"] = <
        text = <"postoperative impaired wound healing">
        description = <"This is a SIMPLE_TEXT object">
    >
    ["at0098"] = <
        text = <"Wound Images">
        description = <"This is a ELEMENT object">
    >
    ["at0099"] = <
        text = <"ED">
        description = <"This is a ED object">
    >
    >
    >
    >
    constraint_definitions = <
    >
    term_binding = <
    >
    constraint_binding = <
    >

```

### 8.1.3 Wound management prescription archetype

```

archetype (adl_version=1.4)
  CEN-EN13606-COMPOSITION.WMPrescription.v1

concept
  [at0000]

language
  original_language = <[ISO_639-1::en-us]>

description
  original_author = <
    ["date"] = <"20110505">
  >
  lifecycle_state = <"Draft">
  details = <
    ["en-us"] = <
      language = <[ISO_639-1::en-us]>
    >
  >
  >

definition
  COMPOSITION[at0000] occurrences matches {1..1} matches { -- WMPrescription
    content existence matches {0..1} cardinality matches {0..*; unordered} matches {
      ENTRY[at0001] occurrences matches {0..*} matches { -- Prescription
        items existence matches {0..1} cardinality matches {0..*; unordered; unique}
      }
    }
  matches {
    ELEMENT[at0002] occurrences matches {0..*} matches { -- Localization
      value existence matches {0..1} matches {
        SIMPLE_TEXT[at0003] occurrences matches {0..1} matches { --
          originalText existence matches {0..1} matches {/.*/}
        }
      }
    }
    ELEMENT[at0004] occurrences matches {0..*} matches { -- Frequency of
      value existence matches {0..1} matches {
        SIMPLE_TEXT[at0020] occurrences matches {0..1} matches {*} -- 1
        SIMPLE_TEXT[at0021] occurrences matches {0..1} matches {*} -- 2
        SIMPLE_TEXT[at0022] occurrences matches {0..1} matches {*} -- 3
        SIMPLE_TEXT[at0023] occurrences matches {0..1} matches {*} -- 4
        SIMPLE_TEXT[at0024] occurrences matches {0..1} matches {*} -- 5
        SIMPLE_TEXT[at0025] occurrences matches {0..1} matches {*} -- 6
        SIMPLE_TEXT[at0026] occurrences matches {0..1} matches {*} --
      }
    }
  }
  ELEMENT[at0006] occurrences matches {0..*} matches { -- Wound Cleansing
    value existence matches {0..1} matches {
      SIMPLE_TEXT[at0007] occurrences matches {0..1} matches { --
        originalText existence matches {0..1} matches {/.*/}
      }
    }
  }
  ELEMENT[at0008] occurrences matches {0..*} matches { -- Debridement
    value existence matches {0..1} matches {
      SIMPLE_TEXT[at0009] occurrences matches {0..1} matches { --
        originalText existence matches {0..1} matches {/.*/}
      }
    }
  }
  ELEMENT[at0010] occurrences matches {0..*} matches { -- Wound Border
    value existence matches {0..1} matches {
      SIMPLE_TEXT[at0011] occurrences matches {0..1} matches { --
        originalText existence matches {0..1} matches {/.*/}
      }
    }
  }
  ELEMENT[at0012] occurrences matches {0..*} matches { -- Wound Specific
    value existence matches {0..1} matches {
      SIMPLE_TEXT[at0013] occurrences matches {0..1} matches { --
        originalText existence matches {0..1} matches {/.*/}
      }
    }
  }
  ELEMENT[at0014] occurrences matches {0..*} matches { -- Compression
    value existence matches {0..1} matches {
      SIMPLE_TEXT[at0018] occurrences matches {0..1} matches {*} --
      SIMPLE_TEXT[at0019] occurrences matches {0..1} matches {*} --
    }
  }
}
short bandage
stocking
}

```

```

ELEMENT[at0016] occurrences matches {0..*} matches { -- Comments
  value existence matches {0..1} matches {
    SIMPLE_TEXT[at0017] occurrences matches {0..1} matches { --
      originalText existence matches {0..1} matches {*}
    }
  }
}
}
}
}
}
}
}
}
}

ontology
terminologies_available = <...>
term_definitions = <
  ["en-us"] = <
    items = <
      ["at0000"] = <
        text = <"WMPrescription">
        description = <"WMPrescription">
      >
      ["at0001"] = <
        text = <"Prescription">
        description = <"This is a ENTRY object">
      >
      ["at0002"] = <
        text = <"Localization">
        description = <"This is a ELEMENT object">
      >
      ["at0003"] = <
        text = <"SIMPLE_TEXT">
        description = <"This is a SIMPLE_TEXT object">
      >
      ["at0004"] = <
        text = <"Frequency of Treatment">
        description = <"This is a ELEMENT object">
      >
      ["at0006"] = <
        text = <"Wound Cleansing">
        description = <"This is a ELEMENT object">
      >
      ["at0007"] = <
        text = <"SIMPLE_TEXT">
        description = <"This is a SIMPLE_TEXT object">
      >
      ["at0008"] = <
        text = <"Debridement">
        description = <"This is a ELEMENT object">
      >
      ["at0009"] = <
        text = <"SIMPLE_TEXT">
        description = <"This is a SIMPLE_TEXT object">
      >
      ["at0010"] = <
        text = <"Wound Border Protection">
        description = <"This is a ELEMENT object">
      >
      ["at0011"] = <
        text = <"SIMPLE_TEXT">
        description = <"This is a SIMPLE_TEXT object">
      >
      ["at0012"] = <
        text = <"Wound Specific Medication">
        description = <"This is a ELEMENT object">
      >
      ["at0013"] = <
        text = <"SIMPLE_TEXT">
        description = <"This is a SIMPLE_TEXT object">
      >
      ["at0014"] = <
        text = <"Compression">
        description = <"This is a ELEMENT object">
      >
      ["at0016"] = <
        text = <"Comments">
        description = <"This is a ELEMENT object">
      >
      ["at0017"] = <
        text = <"SIMPLE_TEXT">
        description = <"This is a SIMPLE_TEXT object">
      >
      ["at0018"] = <
        text = <"short bandage">
        description = <"This is a SIMPLE_TEXT object">
      >
      ["at0019"] = <
        text = <"stocking">
        description = <"This is a SIMPLE_TEXT object">
      >
    >
  >
</term_definitions>

```

```

>
["at0020"] = <
  text = <"1">
  description = <"This is a SIMPLE_TEXT object">
>
["at0021"] = <
  text = <"2">
  description = <"This is a SIMPLE_TEXT object">
>
["at0022"] = <
  text = <"3">
  description = <"This is a SIMPLE_TEXT object">
>
["at0023"] = <
  text = <"4">
  description = <"This is a SIMPLE_TEXT object">
>
["at0024"] = <
  text = <"5">
  description = <"This is a SIMPLE_TEXT object">
>
["at0025"] = <
  text = <"6">
  description = <"This is a SIMPLE_TEXT object">
>
["at0026"] = <
  text = <"daily">
  description = <"This is a SIMPLE_TEXT object">
>
>
>
>
constraint_definitions = <
>
term_binding = <
>
constraint_binding = <
>

```

## 9 References

1. Kozon V, Fortner N. Wundmanagement Pflegephaleristik: ÖGVP Verlag; 2006.
2. Statistik Austria. 2011 [accessed 2011 05/09]; Available from: <http://www.statistik.at>.
3. DNQP. Deutsches Netzwerk für Qualitätsentwicklung in der Pflege: Expertenstandard Pflege von Menschen mit chronischen Wunden. Osnabrück: Fachhochschule Osnabrück; 2009 [accessed 2011 05/09]; Available from: <http://www.dnqp.de/ExpertenstandardChronischeWunden.pdf>.
4. Lichtenstein D. Modernes Wundmanagement: Qualitätsentwicklung in der Pflege von Menschen mit chronischen Wunden [Master thesis]. Vienna: Universität Wien; 2010.
5. Timmermans S, Mauck A. The promises and pitfalls of evidence-based medicine. Health Aff (Millwood). 2005 Jan-Feb;24(1):18-28.
6. Elstein AS. On the origins and development of evidence-based medicine and medical decision making. Inflamm Res. 2004 Aug;53 Suppl 2:S184-9.
7. Openclinical. Evidence-Based Medicine. 2011 [accessed 2011 05/09]; Available from: <http://www.openclinical.org/ebm.html>.
8. Cochrane Collaboration. 2011 [accessed 2011 05/09]; Available from: <http://www.cochrane.org/>.
9. Openclinical. Guidelines. 2011 [accessed 2011 05/09]; Available from: <http://www.openclinical.org/guidelines.html>.
10. Blobel B. Architectural approach to eHealth for enabling paradigm changes in health. Methods Inf Med.49(2):123-34.
11. International Organization for Standardization. ISO/TR 20514:2005 Health informatics -- Electronic health record -- Definition, scope and context 2005.
12. Crossing the Quality Chasm: A New Health System for the 21st Century. Medicine Io, editor. Washington, DC: National Academy Press; 2001.



13. Laleci GB, Dogac A. A Semantically Enriched Clinical Guideline Model Enabling Deployment in Heterogeneous Healthcare Environments. *IEEE Trans Inf Technol Biomed.* 2009;13(2):11.
14. Field MJ, Lohr KN. *Guidelines for Clinical Practice: From Development to Use.* National Academy Press. 1992.
15. Fieschi M, Dufour JC, Staccini P, Gouvernet J, Bouhaddou O. Medical decision support systems: old dilemmas and new paradigms? *Methods Inf Med.* 2003;42(3):190-8.
16. Shiffman RN, Liaw Y, Brandt CA, Corb GJ. Computer-based guideline implementation systems: a systematic review of functionality and effectiveness. *J Am Med Inform Assoc.* 1999 Mar-Apr;6(2):104-14.
17. Boxwala AA, Peleg M, Tu S, Ogunyemi O, Zeng QT, Wang D, et al. GLIF3: a representation format for sharable computer-interpretable clinical practice guidelines. *J Biomed Inform.* 2004 Jun;37(3):147-61.
18. Seyfang A, Miksch S, Marcos M. Combining diagnosis and treatment using ASBRU. *Int J Med Inform.* 2002 Dec 18;68(1-3):49-57.
19. Peleg M, Ogunyemi O, Tu S, Boxwala AA, Zeng Q, Greenes RA, et al. Using features of Arden Syntax with object-oriented medical data models for guideline modeling. *Proc AMIA Symp.* 2001:523-7.
20. Tu SW, Musen MA. Modeling data and knowledge in the EON guideline architecture. *Stud Health Technol Inform.* 2001;84(Pt 1):280-4.
21. Wang D, Peleg M, Tu SW, Boxwala AA, Ogunyemi O, Zeng Q, et al. Design and implementation of the GLIF3 guideline execution engine. *J Biomed Inform.* 2004 Oct;37(5):305-18.
22. Bandini S, Manzoni S, Terenziani P, Montani S, Bottrighi A, Torchio M, et al. Managing Clinical Guidelines Contextualization in the GLARE System. *Advances in Artificial Intelligence: Springer Berlin / Heidelberg;* 2005. p. 454-65.
23. Shahar Y, Young O, Shalom E, Galperin M, Mayaffit A, Moskovitch R, et al. A framework for a distributed, hybrid, multiple-ontology clinical-guideline library, and automated guideline-support tools. *J Biomed Inform.* 2004 Oct;37(5):325-44.
24. Peleg M. Guideline Representation Page - GLIF Specifications. [accessed 2011 05/09]; Available from: <http://mis.hevra.haifa.ac.il/~morpeleg/Intermed/>.

25. Hartvigsen G, Johansen MA, Hasvold P, Bellika JG, Arsand E, Arild E, et al. Challenges in telemedicine and eHealth: lessons learned from 20 years with telemedicine in Tromsø. *Stud Health Technol Inform.* 2007;129(Pt 1):82-6.
26. International Organization for Standardization. ISO 13606-1 Electronic health record communication - Part 1: Reference model. 2008.
27. International Organization for Standardization. ISO 13606-2 Electronic health record communication - Part 2: Archetypes interchange specification. 2008.
28. Gruber TR. Toward Principles for the Design of Ontologies Used for Knowledge Sharing. 1993 [accessed 2011 05/10]; Available from: [http://www-ksl.stanford.edu/KSL\\_Abstracts/KSL-93-04.html](http://www-ksl.stanford.edu/KSL_Abstracts/KSL-93-04.html).
29. Chandrasekaran B, Josephson JR, Benjamins VR. Ontology of Tasks and Methods. 1998 [accessed 2011 05/10]; Available from: <http://www.cse.ohio-state.edu/~chandra/Ontology-of-Tasks-Methods.PDF>.
30. Horrocks I, Patel-Schneider PF, van Harmelen F. From SHIQ and RDF to OWL: the making of a Web Ontology Language. *Web Semantics: Science, Services and Agents on the World Wide Web.* 2003;1(1):7-26.
31. Newsham D, Kalra D, McCay C. HL7 Implementation Guide for 13606. 2006.
32. International Organization for Standardization. ISO 13606 Electronic health record communication - Part 4: Security 2009.
33. Health Level Seven IH. Health Level Seven, Inc. (HL7). 2010 [accessed 2011 05/09]; Available from: <http://www.hl7.org>.
34. Hinchley A. Understanding Version 3 - A primer on the HL7 Version 3 Healthcare Interoperability Standard Mönch, A 2007.
35. Foundation o. openEHR Foundation. 2010 [accessed 2011 05/09]; Available from: <http://www.openehr.org>.
36. Beale T. Archetypes - An Interoperable Knowledge Methodology for Future-proof Information Systems. 2000 [accessed 2011 05/09]; Available from: [http://www.openehr.org/publications/archetypes/archetypes\\_beale\\_web\\_2000.pdf](http://www.openehr.org/publications/archetypes/archetypes_beale_web_2000.pdf).

37. Beale T. Templates and Archetypes: how do we know what we are talking about? 2003 [accessed 2011 05/09]; Available from: [http://www.openehr.org/publications/archetypes/templates\\_and\\_archetypes\\_hoard\\_et\\_al.pdf](http://www.openehr.org/publications/archetypes/templates_and_archetypes_hoard_et_al.pdf).
38. International Organization for Standardization. ISO 13606-3 Electronic health record communication - Part 3: Reference archetypes and term lists. 2009.
39. Blobel B, Engel K, Pharow P. Semantic Interoperability - HL7 Version 3 Compared to Advanced Architecture Standards. *Methods Inf Med*. 2006;45:343-53.
40. W3C. Web Services Architecture. 2010 [accessed 2011 05/09]; Available from: <http://www.w3.org/TR/ws-arch/>.
41. W3C. Simple Object Access Protocol (SOAP). 2010 [accessed 2011 05/09]; Available from: <http://www.w3.org/TR/soap/>.
42. W3C. World Wide Web Consortium (W3C). [accessed 2011 05/09]; Available from: <http://www.w3.org/>.
43. Henning M. The rise and fall of CORBA. *Communications of the ACM* 2008;51(8):52-7.
44. Object Management Group (OMG). Model Driven Architecture. [accessed 2011 05/09]; Available from: <http://www.omg.org/mda/>.
45. Object Management Group (OMG). OMG - Webpage. [accessed 2011 05/09]; Available from: <http://www.omg.org/>.
46. Duftschmid G. Vernetzung des Gesundheitswesens. University Lecture 2010.
47. ISO, IEC, ITU-T. The Reference Model of Open Distributed Processing (RM-ODP). 2010 [accessed 2011 05/09]; Available from: <http://www.rm-odp.net/>.
48. Wikipedia. RM-ODP. 2011 [accessed 2011 05/09]; Available from: <http://en.wikipedia.org/wiki/RM-ODP>.
49. Openclinical. GLIF - Guidline Interchange Format. 2011 [accessed 2011 05/09]; Available from: [http://www.openclinical.org/gmm\\_glif.html](http://www.openclinical.org/gmm_glif.html).

50. Peleg M, Boxwala AA, Ogunyemi O, Zeng Q, Tu S, Lacson R, et al. GLIF3: the evolution of a guideline representation format. Proc AMIA Symp. 2000:645-9.
51. Rebstock M, Fengel J, Paulheim H. Ontologies-Based Business Integration Springer Berlin Heidelberg; 2008.
52. Park C, Shon J. A Study on the Web Ontology Processing System. 2005 [accessed 2011 05/09]; Available from: <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=01462960>.
53. Fayad M, Schmidt DC. Object-Oriented Application Frameworks. Communications of the ACM. 1997;40(10):32-8.
54. Bechhofer S, Carroll JJ. Parsing OWL DL: Trees or Triples? 13nd International Conference on World Wide Web2004. p. 266-75.
55. Cardoso J, Sheth AP. Semantic Web Services, Processes and Applications: Springer Science+Business Media, LLC; 2006.
56. Shearer R, editor. Structured Ontology Format. 3rd OWL: Experiences and Directions Workshop (OWLED2007); 2007.
57. Decker S, Melnik S, Harmelen Fv, Fensel D, Klein M, Broekstra J, et al. The Semantic Web: The Roles of XML and RDF. IEEE INTERNET COMPUTING. 2000:63-74.
58. Bechhofer S, Volz R, Lord P. Cooking the Semantic Web with the OWL API. 2nd International Semantic Web Conference ISWC2003. p. 659-75.
59. Ontotext. OWLIM Semantic Repository, product page. 2011 [accessed 2011 05/09]; Available from: <http://ontotext.com/owlim/index.html>.
60. Kiryakov A, Ognyanov D, Manov D. OWLIM – A Pragmatic Semantic Repository for OWL International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS 2005): Springer Berlin / Heidelberg; 2005. p. 182-92.
61. openRDF. User Guide for Sesame 2.3. 2010 [accessed 2011 05/09]; Available from: <http://www.openrdf.org/doc/sesame2/users/index.html>.
62. Cui Z, Jones D, O'Brien P. Semantic B2B Integration: Issues in Ontology-based Approaches. SIGMOD Record. 2002;31(1):43-8.

63. Yang H, Cui Z, OBrien P, editors. Extracting Ontologies from Legacy Systems for Understanding and Re-Engineering. 23rd IEEE International Conference on Computer Software Applications; 1999.
64. Blobel B, Oemig F. Ontology-Driven Health Information System Architecture. In: Adlassnig K-P, editor. Medical Informatics in a United and Healthy Europe and Healthy Europe: IOS Press; 2009. p. 195-9.
65. Blobel B, Kalra D, Koehn M, Lunn K, Pharow P, Ruotsalainen P, et al. The role of ontologies for sustainable, semantically interoperable and trustworthy EHR solutions. Stud Health Technol Inform. 2009;150:953-7.
66. Yildicz B, Miksch S. Ontology-Driven Information Systems: Challenges and Requirements. International Conference on Semantic Web and Digital Libraries: Indian Statistical Institute Platinum Jubilee Conference Series (2007); 2007. p. 35-44.
67. Hitzler P, Krötzsch M, Rudolph S, Sure Y. Semantic Web - Grundlagen: Springer; 2008.
68. W3C. RDF/XML Syntax Specification (Revised). 2004 [accessed 2011 05/09]; Available from: <http://www.w3.org/TR/rdf-syntax-grammar/>.
69. Hesse W. Ontologies in the Software Engineering process. 2005 [accessed 2011 05/09]; Available from: <http://www1.in.tum.de/static/lehrstuhl/files/teaching/ws0607/Wissensbasiertes%20SE/OntologiesInSE.pdf>.
70. Berners-Lee T, Hendler J, Lassila O. The Semantic Web. ScientificAmericancom. 2001.
71. Mizoguchi R. Tutorial on ontological engineering. 2003 [accessed 2011 05/09]; Available from: <http://www.ei.sanken.osaka-u.ac.jp/pub/all-publications.html>.
72. W3C. RDF Vocabulary Description Language 1.0: RDF Schema. 2004 [accessed 2011 05/09]; Available from: <http://www.w3.org/TR/rdf-schema/>.
73. Object Management Group (OMG). Ontology Definition Metamodel Version 1.0. 2009 [accessed 2011 05/09]; Available from: <http://www.omg.org/spec/ODM/>.

74. Stuckenschmidt H, van Harmelen F. Information Sharing on the Semantic Web: Springer; 2005.
75. Antoniou G, Franconi E, van Harmelen F. Introduction to Semantic Web Ontology Languages. Springer; 2005 [accessed 2011 05/09]; Available from: <http://www.cs.vu.nl/~frankh/postscript/REWERSE05.pdf>.
76. W3C. OWL 2 Web Ontology Language Document Overview. 2009 [accessed 2011 05/09]; Available from: <http://www.w3.org/TR/owl2-overview/>.
77. Bruijn Jd. Semantic information integration inside and across organizational boundaries: TUDelft; 2003.
78. Gómez-Pérez A, Fernández-López M, Corcho O. Ontological Engineering: Springer; 2004.
79. Mika P, Akkermans H. Analysis of the State-of-the-Art in Ontology-based Knowledge Management. 2003 [accessed 2011 05/09]; Available from: <http://km.aifb.kit.edu/projects/swap/public/public/Publications/swap-d1.2.pdf>.
80. DAML. OWL-S. 2010 [accessed 2011 05/09]; Available from: <http://www.daml.org/services/owl-s/>.
81. Martin D, Burstein M, Hobbs J, Lassila O, McDermott D, McIlraith S, et al. OWL-S: Semantic Markup for Web Services. 2008 [accessed 2011 05/09]; Available from: <http://www.ai.sri.com/daml/services/owl-s/1.2/overview/>.
82. Ghallab M, Howe A, Knoblock C, McDermott D, Ram A, Veloso M, et al. PDDL The Planning Domain Denition Language1998.
83. Curbera F, Golland Y, Klein J, Leymann F, Roller D, Thatte S, et al. Business Process Execution Language for Web Services, Version 1.0: IBM.
84. Meseguer J. Conditional rewriting logic as a unified model of concurrency. Theoretical Computer Science. 1992;96(1):73-155
85. Milner R. Communicating with Mobile Agents: The pi-Calculus. Cambridge: Cambridge University Press; 1999.
86. Schlenoff C, Gruninger M, Tissot F, Valois J, Lubell J, Lee J. The Process Specification Language (PSL) Overview and Version 1.0 Specification. Gaithersburg: NISTIR 6459 National Institute of Standards and Technology; 2000.

87. Narayanan S. Reasoning About Actions in Narrative Understanding. International Joint Conference on Artificial Intelligence (IJCAI'1999). San Francisco: Morgan Kaufmann Press; 1999. p. 350-7.
88. Levesque H, Reiter R, Lesperance Y, Lin F, Scherl R. GOLOG: A Logic programming language for dynamic domains. Journal of Logic Programming. 1997;31(1-3):59-84.
89. Martin D, Cheyer A, Moran D. The Open Agent Architecture: A Framework for Building Distributed Software Systems. Applied Artificial Intelligence. 1999;13(1-2):92-128.
90. Finin T, Labrou Y, Mayfield J, editors. KQML as an Agent Communication Language. Cambridge: MIT Press; 1997.
91. Chinnici R, Moreau J-J, Ryman A, Weerawarana S. Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. 2007 [accessed 2011 05/09]; Available from: <http://www.w3.org/TR/wsdl20/>.
92. Pelegri-Llopart E, Yoshida Y, Moussine-Pouchkine A. The GlassFish Community Delivering a Java EE Application Server. 2007 [accessed 2011 05/09]; Available from: <https://glassfish.dev.java.net/faq/v2/GlassFishOverview.pdf>.
93. Wikipedia. Application Programming Interface (API). 2011 [accessed 2011 05/09]; Available from: [http://en.wikipedia.org/wiki/Application\\_programming\\_interface](http://en.wikipedia.org/wiki/Application_programming_interface).
94. Wikipedia. JavaServer Faces. 2010 [accessed 2011 05/09]; Available from: [http://en.wikipedia.org/wiki/JavaServer\\_Faces](http://en.wikipedia.org/wiki/JavaServer_Faces).
95. w3schools. Web Services Tutorial. [accessed 2011 05/09]; Available from: <http://www.w3schools.com/webservices/default.asp>.
96. Bray T, Paoli J, Sperberg-McQueen CM, Maler E, Yergeau F. Extensible Markup Language (XML) 1.0 (Fifth Edition). 2008 [accessed 2011 05/09]; Available from: <http://www.w3.org/TR/xml/>.
97. Wikipedia. Web Service. [accessed 2011 05/09]; Available from: [http://en.wikipedia.org/wiki/Web\\_service](http://en.wikipedia.org/wiki/Web_service).
98. w3schools. WSDL Tutorial. [accessed 2011 05/09]; Available from: <http://www.w3schools.com/wsdl/default.asp>.

99. w3schools. SOAP Tutorial. [accessed 2011 05/09]; Available from: <http://www.w3schools.com/soap/default.asp>.
100. Knublauch H. Ontology-Driven Software Development in the Context of the Semantic Web: An Example Scenario with Protege/OWL. Annex XVII (7): Stanford Medical Informatics, Stanford University, CA; 2004.
101. Devedzic V. Understanding Ontological Engineering. Communications of the ACM. 2002;45(4).
102. Noy NF, McGuinness DL. Ontology Development 101: A Guide to Create Your First Ontology. [accessed 2011 05/09]; Available from: [http://protege.stanford.edu/publications/ontology\\_development/ontology101.html](http://protege.stanford.edu/publications/ontology_development/ontology101.html).
103. Uschold M, Grüninger M. Ontologies: Principles, Methods and Applications. Knowledge Engineering Review 1996;11(2):93--136.
104. Rosch E. Principles of Categorization. Cognition and categorization. 1978:27-48.
105. Knublauch H, Fergerson RW, Noy NF, Musen MA. The Protégé OWL Plugin: An Open Development Environment for Semantic Web Applications. 2004 [accessed 2011 05/09]; Available from: <http://protege.stanford.edu/plugins/owl/publications/ISWC2004-protege-owl.pdf>.
106. Mozilla.org. Mozilla Public License Version 1.1. [accessed 2011 05/09]; Available from: <http://www.mozilla.org/MPL/MPL-1.1.html>.
107. OKBC Working Group. Open Knowledge Base Connectivity Home Page. 1995 [accessed 2011 05/09]; Available from: <http://www.ai.sri.com/~okbc/>.
108. Stanford-University. Protégé-Frames. 2010 [accessed 2011 05/09]; Available from: <http://protege.stanford.edu/overview/protege-frames.html>.
109. Stanford-University. Protégé-OWL. 2010 [accessed 2011 05/09]; Available from: <http://protege.stanford.edu/overview/protege-owl.html>.
110. W3C. OWL Web Ontology Language Guide. 2004 [accessed 2011 05/09]; Available from: <http://www.w3.org/TR/owl-guide/>.



111. Maldonado JA, Moner D, Boscá D, Fernández-Breis JT, Angulo C, Robles M. LinkEHR-Ed: A multi-reference model archetype editor based on formal semantics. *International Journal of Medical Informatics*. 2009;78(8):559-70.
112. LinkEHR. LinkEHR-ED. 2010 [accessed 2011 05/09]; Available from: <http://www.linkehr.com/>.
113. osiris next. OWL-S API. 2010 [accessed 2011 05/09]; Available from: <http://on.cs.unibas.ch/owls-api/index.html>.
114. Open Source Initiative. Open Source Initiative OSI - The MIT License:Licensing. [accessed 2011 05/09]; Available from: <http://www.opensource.org/licenses/mit-license.php>.
115. openEHR foundation. Ontologies Home. 2010 [accessed 2011 05/09]; Available from: <http://www.openehr.org/wiki/display/ontol/Ontologies+Home>.
116. Gerhold L. Demographic Ontology. 2011 [accessed 2011 05/10]; Available from: <http://www.meduniwien.ac.at/user/lukas.gerhold/PhD/ontologies/demographic.s.owl>.
117. Gerhold L. EHR Ontology. 2011 [accessed 2011 05/10]; Available from: <http://www.meduniwien.ac.at/user/lukas.gerhold/PhD/ontologies/ISO13606-EHRATONTOLOGY-v1.1.owl>.
118. Schröder G. *Pflege von Menschen mit chronischen Wunden*. Bern: Hans Huber; 2009.
119. Litzinger G, Rossman T, Demuth B, Roberts J. In-home wound care management utilizing information technology. *Home Healthc Nurse*. 2007 Feb;25(2):119-30.
120. Wilbright WA, Birke JA, Patout CA, Varnado M, Horswell R. The use of telemedicine in the management of diabetes-related foot ulceration: a pilot study. *Adv Skin Wound Care*. 2004 Jun;17(5 Pt 1):232-8.
121. Braun RP, Vecchietti JL, Thomas L, Prins C, French LE, Gewirtzman AJ, et al. Telemedical wound care using a new generation of mobile telephones: a feasibility study. *Arch Dermatol*. 2005 Feb;141(2):254-8.

122. Binder B, Hofmann-Wellenhof R, Salmhofer W, Okcu A, Kerl H, Soyer HP. Teledermatological monitoring of leg ulcers in cooperation with home care nurses. *Arch Dermatol*. 2007 Dec;143(12):1511-4.
123. Ebner C, Wurm EM, Binder B, Kittler H, Lozzi GP, Massone C, et al. Mobile teledermatology: a feasibility study of 58 subjects using mobile phones. *J Telemed Telecare*. 2008;14(1):2-7.
124. Hofmann-Wellenhof R, Salmhofer W, Binder B, Okcu A, Kerl H, Soyer HP. Feasibility and acceptance of telemedicine for wound care in patients with chronic leg ulcers. *J Telemed Telecare*. 2006;12 Suppl 1:15-7.
125. Martinez-Costa C, Menarguez-Tortosa M, Fernandez-Breis JT, Maldonado JA. A model-driven approach for representing clinical archetypes for Semantic Web environments. *Journal of Biomedical Informatics*. 2009;42(1):150-64.
126. Martinez I. Demographic RM. 2005 [accessed 2011 05/09]; Available from: [http://traiano.us.es/~isabel/EHR/Demographic\\_RM.owl](http://traiano.us.es/~isabel/EHR/Demographic_RM.owl).
127. openEHR foundation. openEHR Specifications. 2008 [accessed 2010]; Available from: <http://www.openehr.org/releases/1.0.2/roadmap.html>.
128. Kilic O, Dogac A. Achieving Clinical Statement Interoperability using R-MIM and Archetype-based Semantic Transformations. *IEEE Trans Inf Technol Biomed*. 2008 Jun 10;13(4):467 - 77.
129. Maldonado JA, Moner D, Bosca D, Fernandez-Breis JT, Angulo C, Robles M. LinkEHR-Ed: a multi-reference model archetype editor based on formal semantics. *Int J Med Inform*. 2009 Aug;78(8):559-70.
130. Fernandez-Breis JT, Vivancos-Vicente PJ, Menarguez-Tortosa M, Moner D, Maldonado JA, Valencia-Garcia R, et al. Using semantic technologies to promote interoperability between electronic healthcare records' information models. *Conf Proc IEEE Eng Med Biol Soc*. 2006;1:2614-7.
131. Martinez-Costa C, Menarguez-Tortosa M, Maldonado JA, Fernandez-Breis JT. Semantic Web technologies for managing EHR-related clinical knowledge. In: Wu G, editor. *Semantic Web: INTECH*; 2010. p. 201-18.
132. Entwistle M, Shiffman RN. Turning Guidelines into Practice: Making It Happen With Standards. *Health Care and Informatics Review Online*. 2005.

133. Isern D, Sanchez D, Moreno A. Agents and Clinical Guidelines: Filling the Semantic Gap. Proceeding of the 2007 conference on Artificial Intelligence Research and Development: IOS Press; 2007. p. 67-76.
134. Kumar A, Ciccarese P, Smith B, Piazza M. Context-based task ontologies for clinical guidelines. Stud Health Technol Inform. 2004;102:81-94.
135. Casteleiro MA, Diz JJD. Clinical practice guidelines: A case study of combining OWL-S, OWL, and SWRL. Know-Based Syst. 2008;21(3):247-55.
136. Semantic Health. [accessed 2011 05/09]; Available from: <http://www.semantichealth.org/>.
137. Stroetmann VN, Kalra D, Lewalle P, Rector A, Rodrigues JM, Stroetmann KA, et al. SemanticHEALTH Report. 2009 [accessed 2011 05/09]; Available from: [http://ec.europa.eu/information\\_society/activities/health/docs/publications/2009/2009semantic-health-report.pdf](http://ec.europa.eu/information_society/activities/health/docs/publications/2009/2009semantic-health-report.pdf).
138. Kalra D, Blobel B. Semantic Interoperability of EHR Systems. In: Bos L, Blobel B, editors. Medical and Care Compunetics 4 IOS Press; 2007. p. 231-45.
139. Health Level Seven (HL7). Clinical Document Architecture, Release 2.0 [accessed 2009 06/17]; Available from: <http://www.hl7.org/v3ballot/html/infrastructure/cda/cda.htm>.
140. Grieve G, Townend I, Mulrooney G, Shafarman M, Beeler W, Singureanu I, et al. HL7 Templates. 2009 [accessed 2011 05/09]; Available from: <http://www.hl7.org/v3ballot/html/infrastructure/templates/templates.htm>.
141. WHO. International Classification of Diseases (ICD). 2011 [accessed 2011 05/09]; Available from: <http://www.who.int/classifications/icd/en/>.
142. Logical Observation Identifiers Names and Codes (LOINC®). 2011 [accessed 2011 05/09]; Available from: <http://loinc.org/>.
143. Giannangelo K, Berkowitz L. SNOMED CT helps drive EHR success. J AHIMA. 2005 Apr;76(4):66-7.
144. Ruby. RubyTL - Model Driven Development in Ruby. 2010 [accessed 2011 05/09]; Available from: <http://rubytl.rubyforge.org/>.



# Curriculum Vitae

## Personal Information

---

**Lukas Gerhold, MSc**

Mortaraplatz 11/2/18  
1200 Vienna  
Austria / EUROPE



E-mail [Luke.Gerhold@gmail.com](mailto:Luke.Gerhold@gmail.com); [Lukas.Gerhold@meduniwien.ac.at](mailto:Lukas.Gerhold@meduniwien.ac.at)

Nationality Austria

Date of Birth 27<sup>th</sup> June 1981

## Education

---

- Since 2007 PhD Thesis about "ICT-support for applying evidence based medicine in a distributed environment"
- 1999 – 2007 Study of Medical Informatics at the Technical University of Vienna: Graduation with title Master of Science for Medicine and Computer Science.
- 2007 IBM–Certificate for Project Management
- 2007 Microsoft–Certificate for Open Office XML
- 2006 IBM–Certificate for Web Development

## Professional employment

---

- Since August 2009 Employee at Institute for Information and Retrieval Systems at the Medical University Vienna
- July 2008 – August 2009 Employee at Vienna General Hospital, Medical Directorate
- April 2008 – July 2008 Employee at DSC-GmbH, SAP Consulting and Development
- 2003-2008 Employee at Institute for Information and Retrieval Systems at the Medical University Vienna
- 1997-2003 Employee (part-time job) at Altherm Engineering GmbH.

## Further skills and competences

---

- Project Experience "AKIM" - General Hospital Information Management – Subproject "RDA" Research Documentation and Analysis – Developer and Solution Architect
- "Derma Trials" – Analysis Design and Development of a Trial and Tele-consulting System with Image Support
- "AKIM" - General Hospital Information Management – Design, Definition and Coordination of House-wide Electronical Medical Documentation

"AKIM" - General Hospital Information Management - Project Assistant Manager, Co-Leader of Trial-System Working Group, Group Member of Security- and Privacy- Polices Working Group

"ArchiMed Web" - Clinical Trial and Retrieval System – Developer

"WAF"- Education Platform for Ophthalmology - Developer

"Wamis" and "Wamastat" Migration Project

"Students Administration and Grading System" – Developer

Technical Experience Java2, J2EE, EJB, Web Services, JSP, JSF; PHP, Python, Abap, SmallTalk, Visual Studio .NET (C++, C#), Perl, Ruby;

Db4o, Oracle DB, MySQL, PostgreSQL

Unix, Linux, Win2000 - Win 7, Windows Server 2000-2003, Mac OSX Server

Soft Skills Experienced and Successful as Teamplayer, Creative, Communicative and Humorous

Scientific Experience and Interests Development of ISO13606 Archetypes / HL7 v.3 Templates

EHR – Base Research / Semantic Interoperability / Process Interoperability

Further Development and Review of EHR – Standards

Regular Visits of International Conferences for Health Care Informatics

Ontologies / Knowledge and Information Representation / Semantic Web

Intelligent Systems / Reasoning Systems / Process and Decision Support

Multi-Professional-Health-Care-Management / Telemedicine

Development aid / development cooperation / WHO / UNO

Publications:

1. Gerhold L, Tekoglu H, Dorn M, Binder M. An owl based electronic health record system supporting evidence based medicine. IADIS ICWI 2010 – International WWW Conference
2. Gerhold L, Tekoglu H, Messina LA, Binder M. Evidence based medicine meets electronic health record. CBIS 2010 – Brazilian Congress of Health Informatics
3. Gerhold L, Tekoglu H, Weingast J, Binder M. Dermatrials: an application for implementing telemedical feasibility studies. TeleDerm 2010 – 3rd World Congress of TeleDermatology
4. Gerhold L, Temsch W, Janzek-Hawlat S, Dorn M, Weingast J, Binder M. Advancing quality of health care through ontology based screen representation. TeleDerm 2010 – 3rd World Congress of TeleDermatology
5. Wiesbauer F, Blessberger H, Azar D, Goliash G, Wagner O, Gerhold L, et al. Familial-combined hyperlipidaemia in very young myocardial infarction survivors (< or =40 years of age). Eur Heart J. 2009 May;30(9):1073-9.
6. Gerhold L. Planung und Realisierung eines webbasierten, medizinischen Informationssystems. Publisher VDM. 2008. ISBN 978-3-639-01955-1.
7. Dorda W, Duftschmid G, Gerhold L, Gall W, Gambal J. Austria's path toward nationwide electronic health records. Methods Inf Med. 2008;47(2):117-23.
8. Dorda W, Duftschmid G, Gerhold L, Gall W, Gambal J. Introducing the electronic health record in austria. Stud Health Technol Inform. 2005;116:119-24.
9. W. Gall , G. Duftschmid, W. Dorda, Acknowledgment: Lukas Gerhold. Temporal Components in Architectures of Electronic Health Records, GMDS 2004

Other Experiences Cooperation with Clinical Investigators during Clinical Trials.  
Teamwork with Physicians and Clinical Investigators defining Requirements for the Hospital Information System for the General Hospital of Vienna.  
IT-Support

Hobbies Climbing, Hiking, Mountaineering, Outdoor-geek

Languages German, English, French, Turkish