FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Design and Implementation of an Integration Framework for an Electronic Health Record Based Hospital Information System

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur/in

im Rahmen des Studiums

Software Engineering & Internet Computing

eingereicht von

Klaus Bayrhammer
Matrikelnummer 0525896

an der

Fakultät für Informatik der Technischen Universität Wien

Betreuung

Betreuer/in: Thomas Grechenig

Mitwirkung: Wolfgang Schramm

Wien, _____ _____

(Unterschrift Verfasser/in) (Unterschrift Betreuer/in)

To mom and dad

Declaration of Authorship

I, Klaus Bayrhammer, declare that this thesis titled, “Design and Implementation of an Integration Framework for an Electronic Health Record Based Hospital Information System” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Vienna

.....	.....	.....
Place	Date	Signature

Abstract

The purpose of this thesis was to design and implement an architecture for a hospital information system, which provides an infrastructure for easy integration of external systems.

Because of the complexity of managing information in hospital environments, there are several specialized systems which may have different architectures as well as different scopes of functions and therefore incompatible interfaces. In order to allow those specialized systems to exchange data they have to be integrated into an information system. Therefore, the hospital's core information system has to provide an open architecture which supports integration on a system level. During the course of this thesis an integration framework for a hospital information system has been developed. The system uses persistent one-way messaging which supports loose coupling and complies with the style of service oriented architectures. In addition, the architecture is based on a complex rule based event system which is responsible for the distribution of the messages. This system has been connected to a data storage, which holds patient information based on the OpenEHR standard.

Furthermore, an enterprise resource planning tool as well as a picture archiving and communication system have been integrated as a proof-of-concept. The architecture proved to be effective in integrating third party systems into the hospital's information system. Integrating the ERP system as well as the PACS only required the implementation of a connector plugin and adding or modifying rules in the rule engine. The validity of this project is limited to the use cases which have been implemented in the proof-of-concept, so it would be of further interest whether more complex use cases could be integrated that easily.

Considering the low complexity that is required for integrating systems as well as the technical benefits which come along using this architecture, the usage of service oriented architectures in the health care context seems to be promising. In addition, a message distribution which is controlled by a rule engine enables the system to exchange or integrate new subsystems at runtime.

Kurzfassung

Das Ziel dieser Arbeit war der Entwurf und die Entwicklung einer Architektur für ein Krankenhausinformationssystem, welche einen besonderen Fokus auf die Integrierbarkeit von externen Systemen legt.

In einem Krankenhausinformationssystem müssen viele unterschiedliche Informationen bearbeitet werden. Aufgrund der Komplexität der zu verarbeitenden Informationen gibt es viele spezialisierte Systeme. Diese Systeme basieren oft auf unterschiedlichen Architekturen und werden für unterschiedliche Aufgabenbereiche eingesetzt was zu unterschiedlichen Strukturen und Schnittstellen führen kann. Um diese Systeme zu verbinden und den Datenaustausch zwischen ihnen zu ermöglichen müssen sie in ein gemeinsames Informationssystem integriert werden. Um diese Integration zu unterstützen muss das Krankenhausinformationssystem eine offene Architektur anbieten, welche definierte Schnittstellen für eine Systemintegration bietet.

Im Zuge dieser Arbeit wurde eine Integrationsframework für ein Krankenhausinformationssystem entworfen und entwickelt. Die Infrastruktur basiert auf einem komplexen regelbasierten Eventsystem. Die integrierten Systeme kommunizieren nur über Nachrichten, welche von dem Regelwerk versandt werden und sind so im Sinne einer serviceorientierten Architektur lose gekoppelt. Dieses System wurde an einen Datenspeicher, welcher elektronische Patientenakten mittels dem OpenEHR Standard speichert, angebunden. Zusätzlich wurde ein ERP, sowie ein PACS als Machbarkeitsnachweis in das Informationssystem integriert.

Die konzipierte Architektur erwies sich für die Integration von Drittsystemen als effektiv. Sowohl die Anbindung des ERP Systems, als auch die Integration des PACS erforderten nur die Entwicklung eines Konnektors sowie die Anpassung von Regeln für die Verteilung von Nachrichten. Die Aussagekraft dieses Projekts ist beschränkt auf die Anwendungsfälle welche im Zuge dieser Machbarkeitsstudie umgesetzt wurden. In weiterer Folge wäre es interessant ob auch komplexere Anwendungsfälle mit ähnlich niedrigem Aufwand in das Informationssystem integriert werden können.

Bedenkt man die niedrige Komplexität welche die Integration von externen Systemen mit einer solchen Architektur erfordert, wie auch die technischen Vorteile, welche damit einhergehen so erscheinen serviceorientierten Architekturen im Krankenhausumfeld durchaus vielversprechend.

Acknowledgements

I wish to express my sincere gratitude to my assistant advisor, Dr. Wolfgang Schramm for his inspiration, guidance and continuous support without which this thesis would have never been possible.

I would like to thank my colleagues Michael Fiedler and Harald Köstinger for the invaluable discussions, ideas and for their constructive criticisms.

Moreover, I am profoundly indebted to my parents, my sister and my brother who have always offered their unconditional love, understanding and assistance however demanding my personal and academic endeavors may have been.

Last but not least, I would like to thank Christina who has proved indispensable in every single respect ever since we first met.

Glossary

ADL	Archetype Description Language
AM	Archetype Model
AQL	Archetype Query Language
CDA	Clinical Document Architecture
CEN	European Committee for Standardization
CMET	Common Message Element Type
COTS	Commercial Of-The-Shelf
CPR	Computer based Patient Record
CRM	Customer Relationship Management
DICOM	Digital Imaging and Communication in Medicine
ERP	Enterprise Resource Planning
ESB	Enterprise Service Bus
HIS	Hospital Information System
HL7	Health Level 7
ICD-9	International Statistical Classification of Diseases and Related Health Problems
ICPC	International Classification of Primary Care
IHE	Integrating the Health Care Enterprise
IS	Information System
JMS	Java Messaging Service
LIS	Laboratory Information System

LOINC ..... Logical Observation Identifiers Names and Codes

OCC ..... Original Component Complexes

OpenEHR ..... Open Electronic Health Record

PACS ..... Picture Archiving and Communication System

POJO ..... Plain Old Java Object

RAC ..... Root Architectural Component

REST ..... Representational State Transfer

RIM ..... Reference Information Model

RIS ..... Radiology Information System

RM ..... Reference Model

RMIM ..... Refined Message Information Model

SM ..... Service Model

SNOMED-CT . Systematized Nomenclature of Medicine - Clinical Terms

SOA ..... Service Oriented Architecture

SOAP ..... Simple Object Access Protocol

XML ..... Extensible Markup Language

Contents

Start

CHAPTER 1

Medical informatics: special chapter hospital information system

The main goal of this thesis was to create a proof-of-concept of a future-proof integration architecture for a hospital information system (HIS). If we want to know what a HIS is we have to start by defining an information system (IS) itself.

1.1 Information Systems

An information system is the part of an enterprise that processes and stores data, information and knowledge. More specifically, it can be defined as the socio-technical subsystem of an enterprise which comprises all information processing as well as the associated human or technical actors in their respective information processing roles [1].

"Socio-" describes the human components involved in this system, which could be physicians, administrative staff, IT-staff, researchers, etc. "Technical" refers to the tooling which is used in this information system and can therefore include computers, alongside telephones, scanners or patient records.

If a system involves computer-based information processing it is called a computer-supported information system. An information system can be divided into subsystems which are called

sub-information systems. Each subsystem can carry out specific tasks or provide specific data. Subsystems which use computer based tools are called the computer-supported parts of an information system. The other subsystems are called conventional or paper-based parts [1].

Architecture and Infrastructure of Information Systems

The architecture of an information system describes its fundamental organization, represented by its components, their relationships to each other and to the environment. As the name already suggests it also guides its design and evolution [2]. Basically the architecture of information systems can be summed up by the enterprise functions they provide. This includes the business processes they support and the processing tools which are used together with their relationships to each other [1].

Probably there are multiple different views of an information system. An example would be an enterprise function view which primarily looks at the enterprise functions or a process view primarily looking at the business processes. Architectures which are equivalent regarding specific rules, vocabulary, semantic interpretation and analyses can be summarized in an architectural style [3]. More formally an architectural style characterizes a family of systems that are related by shared structural and semantic properties [4].

Another important term in designing software systems is the infrastructure of a system. The infrastructure describes the information system regarding the types, numbers and availability of information processing tools used in a specific environment or enterprise. [1].

1.2 Hospital Information Systems

With the definition of information systems in mind, it is quite obvious how a hospital information system is defined. A hospital information system is the socio-technical subsystem of a hospital, which comprises all information processing as well as the associated human or technical actors in their respective information processing roles. Considering this definition a hospital has got an information system from the beginning of its existence [1]. Typical parts in a hospital information system are business processes, application components, physical data processing

components and hospital functions.

When new components are integrated into a hospital information system or a new hospital information system is designed from scratch, the staff of the hospital has to be seen as an important part of the information system itself. Therefore it is highly recommended to consider and re-analyze the needs of end-users in every phase of the software's life cycle. [1].

The main goal of a hospital information system is to provide the basis for a well-functioning patient care and patient administration. In addition, the HIS has to support the hospitals economic management and it has to comply with all legal requirements.

To support patient care and administration the common tasks of hospital information systems are [1]:

- Primarily, to make patients information available. This information should be available on time, at the desired location, to authorized staff only and it should be presented in a usable form.
- To make knowledge, e.g. about diseases, about side effects and interactions of medication to support diagnostics and therapy.
- To expose information about the quality of patient care and the economic situation of the hospital.

1.3 Electronic Health Records

When it comes to storing and retrieving a patient's data, hospitals make use of the concept of patient records. A patient record comprise all data and documents, which is collected during a patient's treatment by a health care provider. This data can be stored conventionally (paper-based) or electronically. A patient record contains a set of sub-documentations which have different goals and properties [5].

With this explanation in mind we can go on to the definition of electronic health records. There are several terms which are used instead of electronic health record, like electronic patient record, computerized patient record, electronic medical record, etc. With minor deviations they

all aim for the same target, but there are several different definitions, describing the structure and goals of electronic health records. In the simplest definition the electronic health record is "a computer-stored collection of health information about one person linked by a person identifier" [6]. A more complex definition would be "A computer-based patient record (CPR) is an electronic patient record that resides in a system specifically designed to support users by providing accessibility to complete and accurate data, alerts, reminders, clinical decision support systems, links to medical knowledge, and other aids" [7].

Strengths and weaknesses

The drawbacks of a conventional paper-based patient record are obvious. They are only available at one place at the same time and can be lost. An attribute based retrieval of information is not possible and it cannot be sorted and filtered by user defined criteria. Weed pointed out, that in the current form they are full of serious defects, diffuse, subjective and incomplete [8]. And even worse, over the past few years there had been no improvements [9]. In certain circumstances the integration of different media types is not even possible. As a result there could be more than one patient record existing for one person.

Electronic health records have their strengths where conventional patient records have their weaknesses. They can be accessed on multiple different locations at the same time and it is very unlikely that they get lost. The data can be sorted, filtered and retrieved by user-defined criteria. The integration of different types of media, like lab results or medical images, can be achieved rather easily.

On the other hand, paper-based records can easily be extended by just adding further pages or print-outs to the record. There is no technical equipment required to retrieve data as well as there is no need of technical knowledge to use the system. The paper-based patient record has been used and optimized for decades and therefore clinical staff is used to working with these kinds of records [9].

This is where electronic health records still have their drawbacks. A prerequisite for introducing an EHR to a hospital is, that the hospital is equipped sufficiently. This includes workstations as well as a networking infrastructure. Additionally, the hospital needs to train the medical

staff in using the electronic system.

Common requirements

When designing a electronic health records there are various different requirements that arise from the context of its planned use. First of all there are functional requirements which need to be satisfied. The main functional goal of an electronic health record is to consequently provide non-redundant documentation where stored information can be used by clinical or administrative staff. The literature which describes functional requirements for electronic patient records is extensive [7, 10, 11].

The most important functional requirements for a electronic health records are [11]:

- Use standardized classification systems to assure communication of interpretable medical data.
- Flexibility in respect of the structure of documentation. It has to be easily extensible and reorganizable.
- Flexibility in using the system. This includes the possibility to create custom views to mash up data and to allow free navigation.
- Integration of all required functionalities and data.

But an electronic health record does not only have to provide functional features. There are a lot of requirements concerning usability and presentation of data. This is a crucial success factor for all medical information systems. It includes the possibility of context dependent links, flexible and fast navigation as well as a shallow navigation depth.

Last but not least there are ethical requirements which come along when processing sensitive information. Those ethical requirements include the protection of sensitive data as well as a differentiated security mechanism [9, 12].

1.4 EHR- and related standards

Health care providers all over the globe are faced with the challenge of improving the patient care's quality and efficiency. This can be achieved by shared-care, which means health care providers need to share treatment data. Sharing this data is bound to an extended co-operation and communication of heath care information systems [13]. To assure the semantic interoperability and a seamless patient care over boundaries of information systems, standards need to be defined. Those standards need to describe the terminologies, communication interfaces and technologies as well as a common model for storing and exchanging documents.

HL-7 CDA

"The CDA Release 2.0 provides an exchange model for clinical documents (such as discharge summaries and progress notes) - and brings the health care industry closer to the realization of an electronic medical record. By leveraging the use of XML, the HL-7 Reference Information Model (RIM) and coded vocabularies, the CDA makes documents both machine-readable - so they are easily parsed and processed electronically - and human-readable - so they can be easily retrieved and used by the people who need them. CDA documents can be displayed using XML-aware Web browsers or wireless applications such as cell phones." [1]

HL-7-Clinical Document Architecture (CDA) is a document markup standard which describes the structure and semantics of clinical documents [14]. Key aspects of the HL-7-CDA are:

- CDA documents are encoded in XML.
- CDA documents are based on HL-7-RIM and use HL-7v3 data types.
- CDA documents are expressive and flexible. Templates can be used to constrain the generic CDA specification.

The scope of HL-7-CDA is to standardize clinical documents for the purpose of exchange. The data format outside the exchange context is not specified in this standard. "The CDA does

[1] http://www.hl7.org/implement/standards/cda.cfm, last access: 28-12-2010-15:45

not specify the creation or management of documents, only their exchange markup. While it may be possible to directly use the CDA Schema in a document authoring environment, such use is outside the CDA specification." [14]

A CDA document contains services and observations which have to following characteristics.

- Persistence - An unmodified clinical document has to exist for a time period which is specified by regulatory requirements.
- Stewardship - A clinical document is maintained by a health care provider entrusted with with its care.
- Potential for authentication - Clinical documents need to be legally authenticated.
- Context - A default context for each clinical document is established.
- Wholeness - The authentication of a clinical document is only valid for the whole document in its context. It does not apply to specific parts of the document only.
- Human readability - The clinical document is readable by humans.

The payload of HL-7 CDA documents is solely based on HL-7 RIM classes and HL-7v3 data types.

HL-7 RIM

For assuring interoperability between systems using HL-7v3 standard, all exchanged data has to be based on a common information model as well as a common binding vocabulary. "At the domain-specific level, CMETs, RMIMs, the temporal and procedural conditions expressed by Interaction Diagrams or State Diagrams as well as Application Roles, from which trigger events and interactions result, must be standardized" [13].

Therefore the HL-7 Reference Information Model (HL-7 RIM) has been designed, specifying common data types for health care applications. "HL7's RIM is a comprehensive, non-discipline specific, object-oriented information model of patient care and of the providers, insti-

tutions, and activities involved" [15]. The current HL-7 RIM standard specifies six generic core classes.

- Entity - Any person, material, location or institution
- Roles - The role an entity holds (e.g. physician, patient, ..)
- Participation - The actual role of an entity in a specific act (e.g. patient, witness, ..)
- Act - A health care related activity
- Role Link - A collector class to manage the relationship of entities and their corresponding roles.
- Act Relationship - A collector class for chaining acts.

The concepts of inheritance or specialization (adding characteristics) and cloning (duplicating classes and their characteristics) enables the HL-7 RIM to be tailored to domain specific requirements [16, 17].

Although the HL-7 RIM is a very generic standard, there are some very critical reviews arising, pointing out incoherences in the standard [18].

EN-13606

"The European Committee for Standardization (CEN) is a business facilitator in Europe, removing trade barriers for European industry and consumers." [2]. In 1999 this organization designed the first international architecture of an EHR, the CEN-ENV 13606. According to common CEN guidelines, a CEN ENV has to be reviewed and re-evaluated after three years to cancel, adopt or revise the specification [19]. The currently valid specification, EN-13606, consist of five parts [20]:

- Part 1 - Reference model: A scalable, generic information model to represent health information.

[2] http://www.cexn.eu/cen/AboutUs/Pages/default.aspx, last access 2011-02-21 11:39

- Part 2 - Archetype interchange specification: Archetypes define or constrain legal combinations of the reference model. When communicating between different EHRs an archetype model is used to represent archetypes.

- Part 3 - Reference archetypes and term lists: A basic set of archetypes and terms used in a clinical environment.

- Part 4 - Security: A privilege-based security mechanism to access EHR data.

- Part 5 - Exchange models: This part is still under development, but it will specify the messaging model and the data formats to exchange EHR data.

EN-13606 is solely based on HL-7-RIM, a set of data type definitions harmonized between HL-7, CEN and the EHR Domain Information Model. EN-13606 focuses on structural aspects, which are described through platform independent models [13].

According to EN-13606 an electronic health record comprises two components, Root Architectural Component (RAC) and a Record Component. The Record Component is established by Original Component Complexes (OCC), Selected Component Complexes, Data Items, and Link Items. There have been four components which can be used in the OCC are folders, compositions, headed sections, and clusters. They can be combined recursively to build a hierarchical information model. All components have been defined in one single architectural model, "thus characterizing ENV 13606 as the one-model approach." [13].

OpenEHR

OpenEHR is is an electronic health record standard, which is developed by the OpenEHR foundation. The aim of the OpenEHR foundation is the "development of an open, interoperable health computing platform, of which a major component is clinically effective and interoperable electronic health care records (EHRs)." [21].

The OpenEHR specification describes a standard for storing and sharing EHRs within an open architecture. OpenEHR does that by using a two-level concept. The first level defines a reference model (RM in figure 1.1). This is a general framework, in which all relevant clinical

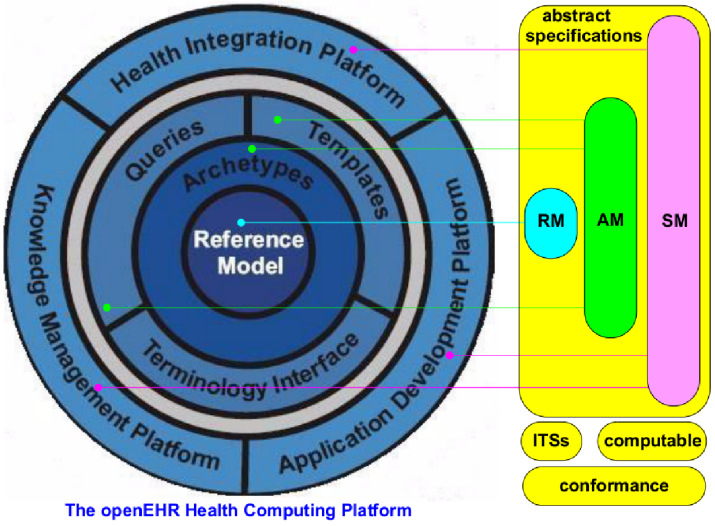

Figure 1.1: OpenEHR Architecture [22]

data can be reliably stored and exchanged. In the second level rules are defined, how to use specific clinical concepts in an electronic health record. These rules, called archetypes (AM in figure 1.1), represent the clinicians' agreed requirements for sharing specific data [23, 22]. The Service Model (SM in figure 1.1) describes a bridge between knowledge resources and information models. The ISO RM/OPD information and computation viewpoints are fully supported in the current OpenEHR release [24].

OpenEHR Reference Model

The reference model in the OpenEHR specification describes the general structure of the electronic health record. It standardizes how contextual information is stored and how clinical information is safely managed and organized. In the reference model each patient has got a single EHR which contains multiple compositions. A composition stores all information based on a specific event, for example a physical examination or a blood sugar measurement. Those compositions can optionally be grouped in folders. Those folders can be used to group information around an episode of care [23, 25].

Like conventional paper-based patient records, compositions contains some kind of head-

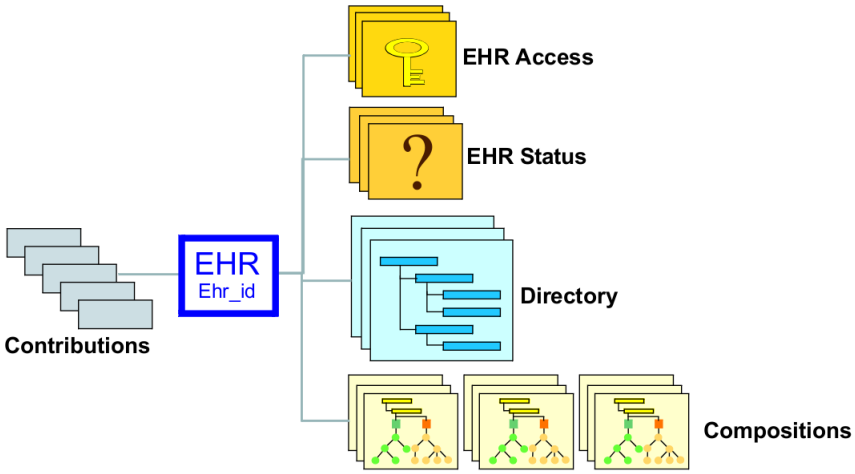

Figure 1.2: OpenEHR high level structures [25]

ings, which are called sections. Those sections can contain one or more entries, which are clinical statements, like a drug, a measurement or a lab result. Each entry can comprise several data items, which are the pieces of information in this entry. For example a medication entry would contain several data items which describe the medication's from, dose and name. Those entries are the most important data types in the OpenEHR reference model, because they define the semantics of all hard information in the record. Furthermore, some meta data concerning the composition is required, like by whom or when it was created. [23, 25].

Beside that, the OpenEHR's reference model specifies functionalities like versioning, labeling of data, auditing, access control, status and control information as well as support for stateful content and record linking [23].

OpenEHR Archetypes

OpenEHR defines rules which describe how the reference model can be used in a certain context and re-used in multiple levels of the reference model. Those rules are called Archetypes [26]. An example for such an archetype would be, for example the patient admission form has to contain an entry about the reason the patient visited the hospital, as well as an entry with the priority with the patients injury. Another example would be a medication archetype which requires a set

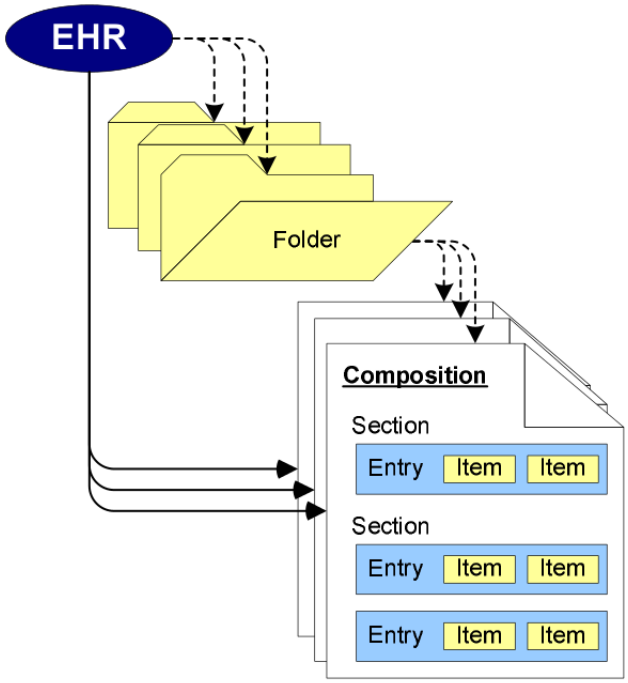

Figure 1.3: OpenEHR reference model [23]

of items, such as an item called “drug name”, which is mandatory and must contain a value from the MIMS medication code set, as well as an item called Form with further properties [23].

From a technical point of view, archetypes use the concept of specialization by restriction. With archetypes it is possible to define constraint patterns on types, attributes, relationships and classes without cloning or renaming as it is required in HL-7v3. To define archetypes an Archetype Description Language (ADL) is used, which provides even more functionalities than just defining constraints. It is specifically designed to integrate established terminology standards and code sets, like ICD-9, ICPC, LOINC or SNOMED-CT [27, 23].

Benefits of using Archetypes

The key benefit of an archetype-based system, like OpenEHR is, the separation of clinical, record management and technical concepts. This supports the users with real clinical experience, because they can focus on the problem of modeling clinical concepts. They know a common

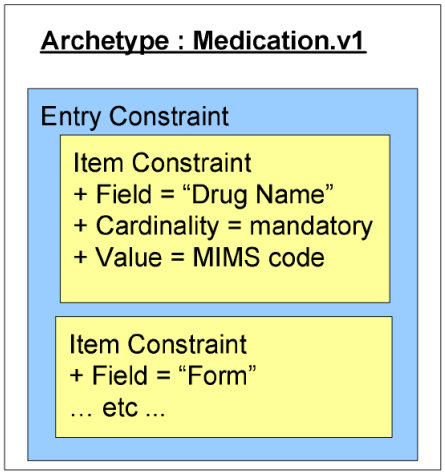

Figure 1.4: OpenEHR sample archetype [23]

framework will take care of the health records interoperability and legally safe records [23].

From a technical perspective the archetype concept makes OpenEHR future-proof and independent of changing clinical knowledge. OpenEHR systems can be implemented based on the OpenEHR reference model by using the standardized methods for storing, querying, updating and exchanging data [26]. New archetype versions, as well as completely new archetypes can be added to the system at runtime, so there are no downtimes required while the system evolves [23].

OpenEHR Versioning

The versioning of information in electronic health records is an integral feature. Because of that, the requirements for versioning are specified in detail in the OpenEHR standard. The standard requires versioning for selected top-level elements, like EHRs and compositions only. OpenEHR heavily relies on the concepts of change-set based versioning and the virtual version tree [22].

The change-sets in the OpenEHR context are called contributions, which consist of new or changed items in the repository. From a technical point of view, the contributions act like transactions. Storing a contribution brings the repository from one consistent state into another consistent state. This is necessary, because arbitrary combinations of changes to single controlled

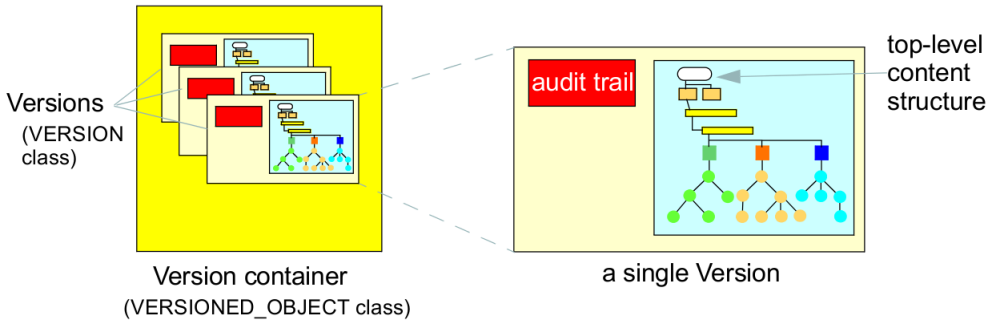

Figure 1.5: OpenEHR version control structures [22]

items could lead to an inconsistent state which is dangerous when clinical data is involved[22].

OpenEHR uses the concept of a virtual version tree as an underlying versioning model. In this design approach data is commited into a repository in lumps, each lump being the data of one version. Each version of an object has its place within the version tree which is managed by a version object. Furthermore, the design guarantees, that no matter where data is created , there are no inconsitencies due to sharing. So OpenEHR is fully capable of sharing data in a shared care context. [22].

OpenEHR and Interoperability

There are various reasons why development and adoption of both international and national standards for interoperability is important in the context of electronic health records [28]:

- Sharing patient information between health care professionals in a multi-disciplinary field.
- Interoperability within organizations of an enterprise or a region.
- Supporting compatibility and interoperability between software from different vendors.

OpenEHR supports exchanging patient data encoded in the XML format. These messages can be easily interpreted using any standard toolkits. Furthermore, OpenEHR is compliant with various established standards. For example the EN-13606 specification is a complete subset of the OpenEHR specification. Moreover is compatible with HL-7v2 and the HL-7 Clinical

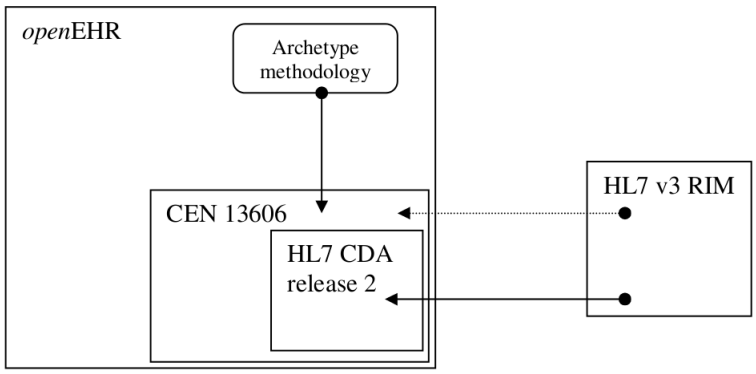

Figure 1.6: Relationship of OpenEHR, CEN-13606 and HL-7-CDA [28]

Document Architecture (CDA), whose messages can be translated into OpenEHR data. So the OpenEHR specification provides both, functional interoperability, as well as semantic interoperability [28, 23].

DICOM

The Digital Imaging and COmmunication in Medicine (DICOM) is the most established international standard for digital medical imaging. The DICOM standard specifies all necessary requirements for diagnostically accurate presentation and processing of medical images. But DICOM is more than only a file format. It is a data transfer, display and storage protocol which aims for covering all requirements which arise when working with digital medical imaging [29]. The DICOM standard has got a major role when it comes to digital medical imaging and has left a mark on contemporary medicine by providing [30]:

- A universal standard of digital medicine. All standard medical imaging devices produce DICOM images and communicate by using the DICOM protocol. Furthermore, health care work-flows are implicitly constrained by a set of DICOM rules.
- Excellent image quality. DICOM uses the most advanced image representation techniques and for example provides up to 16 bits of shades of gray for monochrome image display [31, 30].

- Full support for numerous image-acquisition parameters and different data types. DICOM stores a lot of image related parameters, such as size of the object, slice thickness or a patients 3D position. This information enriches the image and allows better processing and interpretation of the data [32].

- Complete encoding of medical data. DICOM encodes more than 2000 standardized attributes (based on the DICOM data dictionary) to convey meta-information about the image, like a patients name or its diagnostics.

- Clarity in describing digital imaging devices and their functionality. The DICOM standard describes the required functionality for a digital imaging device in a strict but device independent way. This minimizes the space for errors.

The latest specification of the DICOM standard can be found on the DICOM homepage [3] and is publicly available. Currently the DICOM standard holds 16 parts (volumes 1-18, with 9 and 13 being retired). The last publicly available revision, released in 2009, was used.

IHE Profiles

Standards often describe systems more abstract than engineers would need to design and build a system. This always leaves some space for the engineers interpretation. Because of that, it always requires a major effort to connect different systems in the health care context, even if they all comply with international standards [33]. There is no clear guideline on how to use the information provided in the standard to solving specific clinical problems. This leads to gaps between the formal specification of standards and the actual implementation of the standard [34].

The integrating the health care enterprise (IHE) initiative was founded to fill this gap. The IHE process gives health care providers a way to communicate their integration requirements in order to guarantee an optimal patient care. Based on these needs, representatives of information systems and imaging companies build and document a reference implementation of established standards to provide the needed functionality. Their selections are stored in the IHE technical framework [33].

[3] ftp://medical.nema.org/medical/dicom/2009/, last access 29-12-2010 11:31

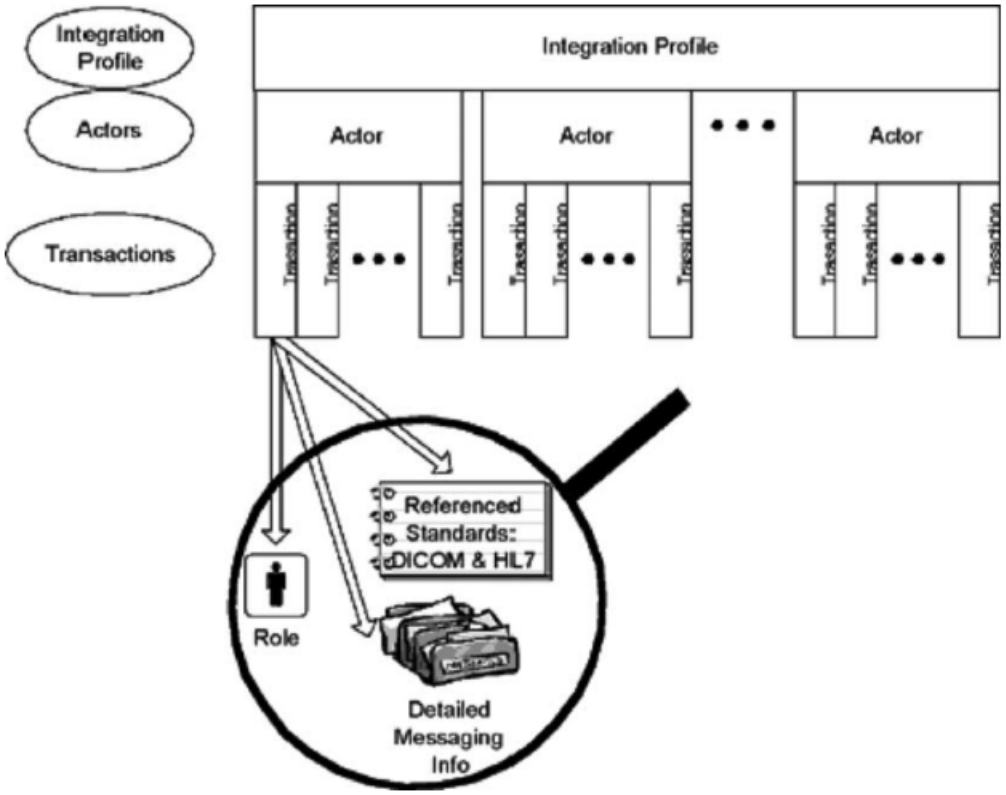

Figure 1.7: IHE Technical Architecture [33]

The IHE technical framework is set of road maps on how to apply established standards to solve system integration problems. This framework uses the DICOM and the HL-7 standards to describe specific integration solutions. The IHE technical framework makes use of following or defines following concepts:

- A data model which shows the relationships between key frames. This data model is based on the data models of DICOM and HL-7.
- The concept of IHE actors. Those actors are used to describe a systems in enterprises in generic, product-neutral terms. Actors exchange messages in order to execute specific tasks.
- Integration profiles are the discrete unit of functionality. Those integration profiles address

a specific clinical and may require several actors and transactions. Currently there are 11 IHE integration profiles defined [4].

1.5 Selected highly relevant components in a Hospital Information System

In most cases a hospital information system is no big monolithic application which provides all the required functionalities. As mentioned in 1.2 there are a lot of different functional requirements for a HIS. The common architecture for a HIS therefore is, to define leading systems for a specific group of requirements, e.g. to use picture archiving and communication systems for medical imaging. Those systems will be integrated into the HIS' core system by the standardized interfaces they provide. In this section we will take a closer look at enterprise resource planning tools and picture archiving and communication systems-

Enterprise Resource Planing tools

Hospitals are complex, information intensive systems which are community-served and quality-focused, which differs them from other types of organizations [35]. But like other types of organizations, the limitation of resources such as budget, personnel, and facilities forces health care organizations to learn to manage their operations efficiently to overcome these constraints. Therefore information technology, especially enterprise resource planning (ERP) tools have proven to be a key component in enhancing an organization's efficiency.

Enterprise resource planning (ERP) tools are packaged application software solutions which seek to integrate the complete range of business processes and functions in order to present a uniform view of the business from a single information and IT architecture [36]. ERP tools tend to be very generic and therefore highly configurable. This gives them the ability to accommodate at least the basic needs of most sectors of the economy. But when it comes down to using ERP systems in the health care sector bigger vendors like Oracle [5] or SAP [6] try to provide customized

[4] http://wiki.ihe.net/index.php?title=Profiles, last access 28-12-2010 21:13
[5] http://www.oracle.com/us/industries/healthcare/index.html, last access 2011-03-02 01:51
[6] http://www.sap.com/industries/healthcare/index.epx, last access 2011-03-02 01:51

health care solutions.

For health care providers operations can be classified into key functions including finance and accounting, logistics and supply chain management, facilities management, human resources, business planning and performance improvement and information management [35]. What those operations have in common is, that they all create a huge amount of data, which needs to be processed. This is where ERP-tools come into play. They can support standardized processes by supplying necessary data or directing users through well defined work-flows [37].

In the past, many hospitals tried to introduce information-technology systems, like ERP-tools [38] or electronic medical records, in order to improve the quality of their service or provide real-time resource monitoring. In general ERP-tools provide various functionalities, such as customer relationship management (CRM), project and quality management or business intelligence reports [37]. But in the health care environment, human resources, supply chain management and financial and accounting functionalities tend to be of predominant importance [35].

Although ERP-tools are getting popular more and more, and the range of functions they provide gets bigger and bigger, many ERP adoption projects were not successfully finished or ended up taking a long time for implementation and integration into the firms business process [37]. Many different factors need to be considered when adopting ERPs. For example organizational characteristics like structure, size, and culture has been recognized as important factors affecting ERP adoption [39]. The integration of the ERP system could also change some of the companies characteristics. For the organization it is necessary to know how to handle these changes so they won't cause problems, like resistance from users, which is one of the key concerns in ERP adoptions [40].

Health care providers have been known to be very engaged in new medical technologies, but considered as a late adopter in terms of integrating information technologies or supporting business operation by information technologies [35]. In the past IT in hospitals was only used for billing and financial activities [41]. But the trend goes clearly to supporting more complex and complicated processes using IT. This should help hospitals to reduce costs, use their resources more efficiently and last but not least meeting the patients expectations.

Even though a lot of hospitals use ERP systems now, it is rather rare, that this ERP system is

the only information system in the hospital, nor are they using all of the ERP's functionalities. This is explained by the unique character of the health care context [35]. Typically ERP systems are not capable of handling all hospital related processes. Therefore specialized systems, like picturing archiving and communication systems (PACS) or decision support systems are needed. Several health care organizations adopt the "best of breed" concept which considers specialized systems for different operations [37]. Most these "best of breed" approaches include an in-house ERP system which is in charge of material management, financial and accounting tasks or other health care operations.

Picture Archiving and Communication Systems

Every state-of-the-art digital radiology department is supported by two main computer systems. First the Radiology Information System (RIS) which is in charge of all text-based functionalities, like material management, billing or shift plans. This system cooperates closely with a Picturing Archiving and Communication System (PACS). The PACS provides all image-related functionalities, such as storage, acquisition or local distribution [42].

A PACS integrates many components which are needed for medical imaging in clinical practice. Depending on the context, a PACS can either comprise just a few components communicating with each other or be a hospital-integrated or enterprise system [43]. Today most of the PACS installed are large-scale PACS according to the conditions specified by Baumann in 1996 [44, 45].

During the time PACS systems evolved three architectural styles established, which are mainly used. These three styles are:

- Stand-alone
- Client-Server
- Web-based

Regardless the architectural style the PACS is based on, there are three major components in a PACS. The image and data acquisition gateway, a PACS server and archive and display

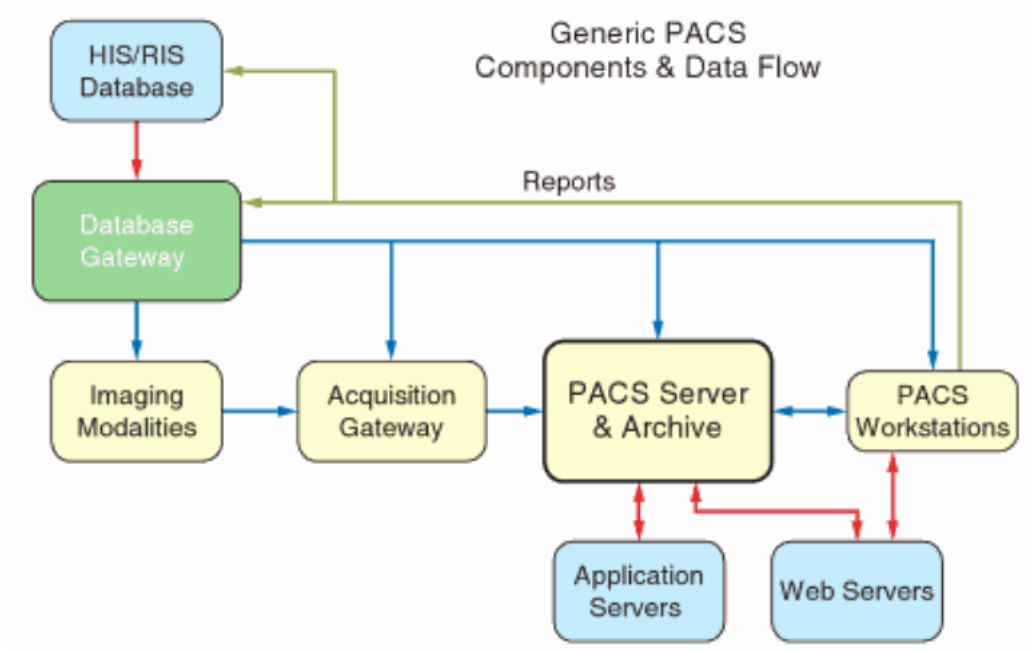

Figure 1.8: PACS architecture [43]

workstations integrated in a digital network [43].

The PACS retrieves images from imaging devices (modalities) and enriches them with textual data from the hospital information system or the radiology information system. There are two sorts gateways the PACS server and archive retrieves its data from. First the database gateway which provides the textual data, like patient's name or diagnostics, and secondly the image acquisition gateway. The image acquisition gateway delivers the image data from the imaging devices. A major task in a PACS is the acquisition of images in reliably and timely manner through the imaging acquisition gateway and to enrich this image with textual data from the database gateway. The acquired images of the current imaging device are encoded and exchanged using the DICOM standard [43].

"The four major ingredients in the PACS infrastructure design concept are system standardization, open architecture and connectivity, reliability and security"[43]. All current PACS systems aim to support as many established industrial standards as possible. For example images are exchanged using the DICOM standard, textual data is stored in ASCII Text, IHE profiles

describe the work-flows or XML is used to represent and spread the data in the Internet. Besides that, it is crucial for a PACS to provide standardized interfaces, so other components can retrieve data regardless the actual implementation of the PACS. Due to the rapid change of technologies most PACS are based on an open and extensible architecture. Furthermore, the reliability of PACS is very important because of two reasons. Firstly, a PACS consists of multiple components, so the probability of one failing is high. Secondly a PACS handles critical patient data, so long downtimes cannot be accepted. Therefore fault tolerant measures like error detection, intelligent software recovery or hardware redundancy is used. Last but not least the patient's data is very sensitive and has to be protected against unauthorized access. The security concept in a PACS has to deal with misuse of data as well as behavioral violation and physical intrusion [46].

1.6 HIS Security

Dealing with sensitive, personal medical data, health information systems need to meet comprehensive security requirements. Regarding security concepts in general, we have to look for security, safety and quality [47].

Security and privacy contain a wide range of aspects, political as well as social, organizational and technical. Establishing a secure infrastructure where an information system can be built on is a highly complex task. As a result many recommendations and best practices evolved. The ISO/IEC Standard 17799 "Information technology - Code of practice for information security management" defines a the common guidelines an information system should comply with. Furthermore, the ISO/IEC IS 15048 "Information Technology - Security Evaluation Criteria" (known as Common Criteria) provides common security evaluation criteria [48].

The standards mentioned before are domain independent so they have to be supplemented with the EU directives 2007/47/EG (known as medical devices directive) which defines rules a medical device has to comply with.

Security Risks and Threats to Clinical Information

The main security threat within a hospital information system comes from abuse by insiders. The effects on aggregating the personal medical data is foreseeable. The likelihood that information will be improperly disclosed depends on its value, and the number of people who have access to it. The aggregation of this information increases both these risk factors [49].

For obvious medico-legal reasons, integrity and availability of hospital information systems are crucial. Like phone, mail or fax messages software systems are prone to failures. But in contrast software bugs may not be as evident as the failure mode of a telephone or fax. Software bugs could alter laboratory results or reports without changing it too much that it would be rejected [50].

Beside the random software bugs there are malicious failures as well. Especially in a shared care environment it may be possible for outsiders to intercept or modify messages. But the majority of attacks on system integrity is carried out by insiders. Typical cases are attempts to shift liability by altering records of malpractice, to abuse prescription systems or straightforward theft or fraud by changing records of stocks or contracts [50].

In addition, there are system level effects as well. For instance attacks on the integrity may be more likely by the loss of confidentiality. If, as in the USA, medical records become widely used outside of clinical practice for hiring and credit decisions, then there will be motives to alter them [50].

However, the biggest problem may be if patients loose the trust in the confidentiality of their personal medical data. If they do, they will suppress relevant information, which results in inaccurate records, as well as poor treatment and to an increased risk to others from spreading infectious disease [51].

Security Policy Model for Clinical Information Systems

A security policy model is a scheme for specifying and enforcing security policies. A security policy model for clinical information systems comparable to the Bell-LaPadula model for military systems [52] and the Clark-Wilson model for banking systems [53] has been proposed by

Ross Anderson [51]. This policy model is based on the rule set rocemmended by the General Medical Council [7] and the British Medical Association [8].

This model defines nine security policies and principles:

Access control lists

Each identifiable clinical record shall be marked with an access control list containing the list of people which should be able to access the record in any way. It should prevent the others to access the record at all.

Record opening

This security policy model avoids multilevel objects and recommends multiple records. Because of that, clinicians have to be able to open records with themselves and the patient on the access control list.

Control

A single clinician has to be marked as being responsible for the patient. This clinician alone is able to add or remove health care professionals to the access control list.

Consent and notification

The patient must approve a modification on an access control list of his medical record. The only exceptions to this rule are in case of an emergency or of statutory exemptions.

Persistence

There rules on how long to medical record have to be kept at least. No one should have the ability to delete clinical information until the appropriate time period has expired.

Attribution

The whole access to clinical information has be marked on the record with the subjects name, date and time. Furthermore, an audit trail must be kept of all deletions.

Information flow

When there are multiple records for a single patient with different access control lists, the information flow is only allowed from the less to the more sensitive record.

Aggregation control

[7] http://www.gmc-uk.org/, last access 2011-05-16 23:34
[8] http://www.bma.org.uk/, last access 2011-05-16 23:33

Clinicians in charge of a safe-haven may be added to a large number of access control lists, making them vulnerable to inducements. In order to prevent this, patients have to be notified if any person whom it is proposed to add to their access control list already has access to health information of a large number of people.

Trusted Computing Base

All computer systems which handle sensitive, personal medical information shall have a subsystem that enforces all principles effectively.

CHAPTER 2

Problem statement and basic idea

The main goal of this thesis is to do a proof of concept for a future-proof integration architecture for a hospital information system. To be able to design and implement such a system it is absolutely necessary to understand the problems current HIS are confronted with.

2.1 Problems of current HIS

Although there has been a reformation in the recent HIS development, many of those systems are still management-centered systems. To improve the quality of the patient's care in a hospital, those systems should be patient-centered clinical systems. In consequence, electronic health records should be the basic design concept [54].

As already discussed in 1.4, the first international approaches to standardize electronic patient records are relatively new. In contrast, computer based hospital information systems are used for decades. Because of that, it is unlikely for older HIS to support, standards which have been released recently. In addition, the implementation is normally steered by software companies, who build independent systems, which tend to lack interoperability [54, 55].

Another urgent problem is the integration of specialized systems, within the hospitals environment. Normally a HIS is no big monolithic software package. There is a core HIS, which integrates several specialized systems, e.g. PACS, RIS or ERP-tools [56]. Those tools

are defined as the leading systems for specific tasks and operations, but they have to provide interfaces to trigger processes or provide data for the core hospital information system. Furthermore, they have to retrieve data from the HIS as well as triggering work-flows in the core system [57, 58, 59].

All in all there is a great need for interoperability for systems in a health care environment. This does not only include sharing information between hospital information systems, but as well communicating information between the different software components within a HIS [58].

2.2 Service oriented and patient centered approach

To solve the problems stated above the architecture of a hospital information system has to make integration of subsystems as easy as possible. Further on, the data has to be stored patient-centric and established standards, such as EN13606, HL-7, DICOM or IHE Profiles have to be supported.

To accomplish the interoperability of different subsystems, the architectural style used, is a service oriented architecture. By using this style different components act as services which only communicate via defined interfaces. This implies a loose coupling of components which gives the maximum flexibility when changing implementations of components which are already used, as well as integrating new software packages [58].

To comply with the requirements of a patient-centric system, the OpenEHR standard is used as an electronic health record implementation. As stated before (see 1.4), EN-13606 is a complete subset of the OpenEHR specification and the archetyped concept of OpenEHR seems promising because it is extensible, as well es vendor-independent.

2.3 Related work

A research project which is based on OpenEHR as well is Opereffa [1]. The Opereffa project is a prototype project for implementing the OpenEHR standard. In contrast to the practical

[1] http://opereffa.chime.ucl.ac.uk/, last access 2011-01-14 21:13

part of this thesis, Opereffa uses a relational database (MySQL [2]) for storing and retrieving the OpenEHR data. Further on, Opereffa does provide a graphical user interface which is dynamically rendered according to the rules specified by the OpenEHR archetypes and templates.

In addition, the OpenEHR research community is very active and some countries like Australia or Sweden invest heavily in the OpenEHR development. Because of this there has been a steady development of the OpenEHR standard, but unfortunately a lack in practical applications.

There has also been research towards the use of service oriented hospital information systems. In [55] the problem of integrating distributed health care information systems is described. A possible solution has been suggested in [56] where a SOA enabled Health Information Integration Platform has been designed. The same topic is addressed by [59] where a more general approach of how to use service oriented architectures in health care information integration environment is described. In addition, [58] discusses the implications of service oriented architecture for large scale distributed health care enterprises. More focusing on technical aspects, [60] discusses the use of webservices in the health care context.

[2] http://www.mysql.com/, last access 2011-02-07 21:54

Middle

CHAPTER 3

Evaluating Of-The-Shelf Software For Extending a HIS-Core System

Many large scale systems depend on commercial of-the-shelf software (COTS). This type of software is evolved, supported by a vendor and used without modification of the internals. Enterprises can get many benefits when using of-the-shelf software. For example, improvements of the performance are made within the product. Moreover it provides the chance of a faster delivery for end-users as well as sharing development costs of new features with other users of the software [61].

Besides those benefits, the risk of vendor or system lock-in arise when using commercial software. An organization locked into a vendor relies on its service, products and updates. Switching to another vendor would entail significant costs. The use and adoption of open source software can reduce the risk of vendor lock-in as well as the dependency to specific vendors [62].

When designing large scale systems, which depend on so called COTS software, the evaluation as well as the selection of those products is a critical factor for the success of the project. Yet many organizations struggle with the process of evaluation and selection. Given that problem some methodical evaluation processes evolved. Two very popular methods are the DESMET software evaluation [63] and Process for COTS Software Product Evaluation [61].

3.1 DESMET Project

"DESMET was a collaborative project part- funded by the UK Department of Trade and Industry which aimed to develop and evaluate a method for evaluating software engineering methods and tools." [63]. The focus of the a DESMET evaluation is to evaluate specific methods or tools for a particular organization. An organization can either be a company as well as a research group. The DESMET method provides guidelines for an evaluator to plan and execute an evaluation. The result of this evaluation is unbiased and reliable, which increases the chance of choosing the tool which suits the best.

DESMET evaluations are context-dependent. The tool which is determined the best in a certain evaluation is the best for the given circumstances. So it may be that an evaluation may come to a different result, if it is executed in two different companies. This is because the context in those two companies may be different. However, DESMET describes one evaluation strategy which is based on a formal experiment [63]. This kind of evaluation is context-independent, because of the context-independent nature of formal experiments [64].

A prerequisite which is formulated in the DESMET evaluation is, that the object which should be evaluated is one of the following [63]:

- A generic method which is a generic paradigm for some aspects of software development.
- A method which is a specific approach within a generic paradigm.
- A tool which is a software application that supports a well-defined activity.

Furthermore, DESMET defines two types of evaluation strategies which aim for different goals. The quantitative evaluation can be used when you want to determine the expected benefit of a new method or tool in measurable terms. On the other hand, the qualitative evaluation assesses the appropriateness of a method or a tool. It can be measured in features provided, the characteristics of its supplier or the training requirements [63].

3.2 DESMET Feature Analysis

In the course of this thesis an open source ERP system (see 6) as well as an open source PACS (see 7) have to be evaluated in respect of their integration capabilities. DESMET describes various ways of evaluating tools or methods. For each of them there are recommendations given when to choose the specific method. For the evaluations which had been done during this thesis, a large number of tools could be possible fits. Those tools have to be evaluated in a short period of time. In respect of those two conditions, the Feature Analysis - Screening mode had been chosen.

This evaluation strategy is based on reviews of marketing and technical documentation rather than trials of the tools themselves. In the simplest form a feature analysis is a list of "yes/no" questions. When it comes to evaluating complex software tools, the requirements become fuzzier. Some products meet certain aspects of a requirement. It that case more sophisticated scoring systems come into play. In such cases you have to provide scores for the degree the feature is provided.

The process of evaluating a specific tool can be broken down into nine steps [63]:

1. Select a set of candidate method/tools to evaluate
2. Decide the required properties or features of the item being evaluated.
3. Prioritize those properties or features with respect to the requirements of the method/tool users.
4. Decide the level of confidence that is required in the results and therefore select the level of rigor required of the feature analysis.
5. Agree on a scoring/ranking system that can be applied to all the features.
6. Allocate the responsibilities for carrying out the actual feature evaluation.
7. Carry out the evaluation to determine how well the products being evaluated meet the criteria that have been set.

8. Analyze and interpret the results.

9. Present the results to the appropriate decision-makers.

CHAPTER 4

EHR core system

As already mentioned above, we want to use a service oriented approach for connecting the different components of a hospital information system. This implies, that each service can be deployed independently from each other and that each service provides an interface for other services to consume.

Considering this, the first service which has been developed was the EHR core system. The requirements for this EHR core system are quite simple. It has to store all patient-related data in a patient-centered, standardized data model. Furthermore, it has to provide versioning and archiving features. According to the architecture, the EHR core system has to expose CRU(D) functions for patient related data via webservices.

4.1 Underlying standard - OpenEHR

First of all, an underlying standard has to be selected, which can be used to store patient information. As already mentioned before, there are various standards describing different concepts of electronic health records, e.g. OpenEHR (see 1.4) or EN-13606 (see 1.4).

For this proof of concept, the OpenEHR standard has been selected, because of multiple reasons [21].

- OpenEHR is a standard, which is actively developed by a transparent foundation, the OpenEHR foundation.
- OpenEHR complies with the EN-13606 standard, because EN-13606 is a complete subset of OpenEHR.
- OpenEHR has an archetype-based architecture, and therefore separates clinical knowledge from the technical infrastructure. Furthermore, the model of clinical concepts can be extended by just defining new archetypes and templates in a domain specific language. A technical framework will take care of interoperability, usability and legally safe records.
- OpenEHR provides a reference implementation, developed in the Java programming language, which contains basic functionalities for the reference model as well as the archetype model.

In addition, there aren't many HIS, which are based on OpenEHR. So the practical part of this thesis can be seen as a further proof of how effective the OpenEHR standard can be used.

4.2 Architectural decisions

Because of the service oriented architecture used in the hospital system itself, the EHR core system has to provide a webservice or RESTful service interface. This is the only architectural requirement, which has to be satisfied in order to comply with the HIS' architecture.

To design the internal architecture of the EHR core system in a future-proof way, a multi-layered approach has been chosen. Three layers have been used encapsulate the different functionalities. Those three layers are

- A model layer, which is responsible for storing, retrieving and versioning all patient-related data.
- A service layer, which holds the business logic of this component.
- A webservice layer, which exposes SOAP webservices and RESTful services.

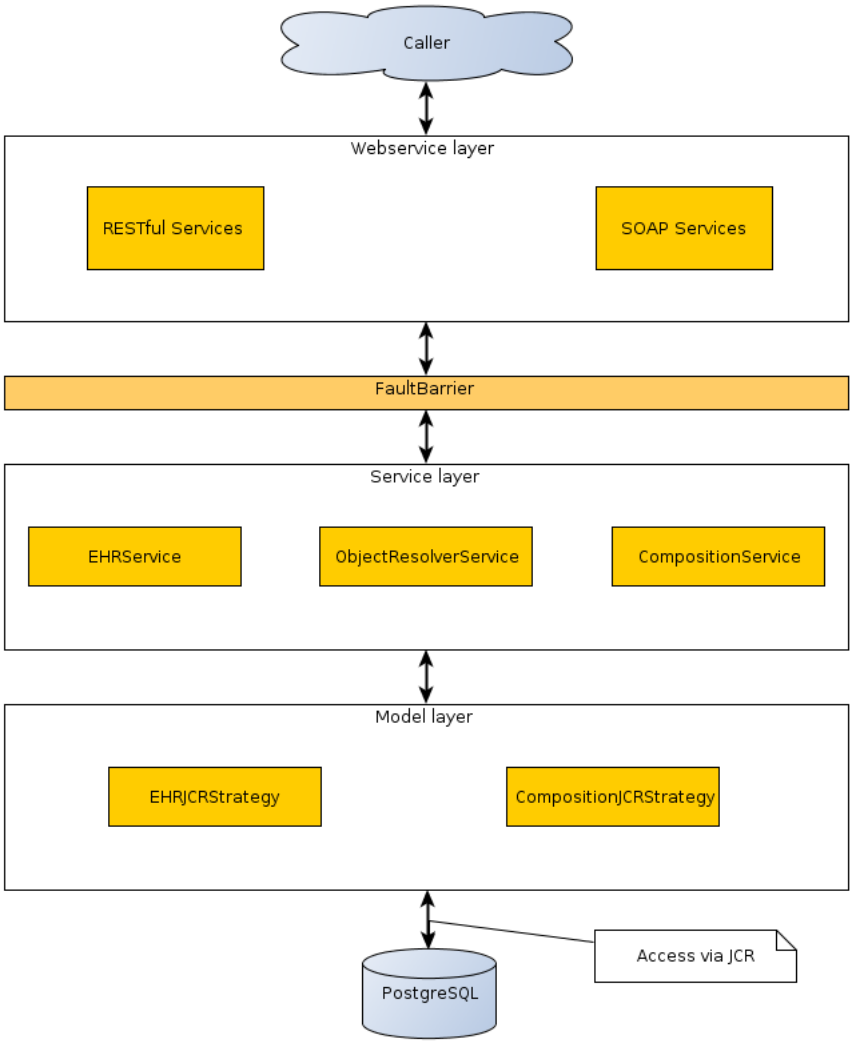

Figure 4.1: EHR core system architecture

Those three layers, as well as the concepts and frameworks used, will be described in the following sections.

Model layer

The most important requirement for the model layer is to conform with the data model specified by the OpenEHR standard. This data model is designed hierarchically and supports an archetype concept. It also specifies requirements for archiving and versioning.

Because of the size of OpenEHR standard's data model, the model layer of the EHR core system only supports a subset of all components specified. The supported components are:

- EHR root objects - This object relates to a single patient and is immutable after its creation. It is the entry point for component parts of the EHR.
- Compositions - A contribution is a "unit of information corresponding to the interaction of a health care agent with the EHR" [25].
- Sections - Those sections can be seen as some kind of heading in a composition. This sections define a logical structure for authors, who creates and updates records, as well as a navigational structure for readers, who consume existing data.
- Entries - An entry is a single "clinical statement", which can be a single sentence, but also a microbiology result.

The composition, as well as sections and entries are archetpyed, which means they are defined by archetypes (see 4.2).

After the set of supported components has been defined, a tool or framework has to be selected. There are various choices of how to store and retrieve information in state-of-the-art software systems.

The most popular approach is using a relational database. Opereffa [1], another project using OpenEHR as a basis to implement a HIS, uses a relational database (MySQL [2]) to store

[1] http://opereffa.chime.ucl.ac.uk/, last access 2011-01-14 21:13
[2] http://www.mysql.com/, last access 2011-02-07 20:56

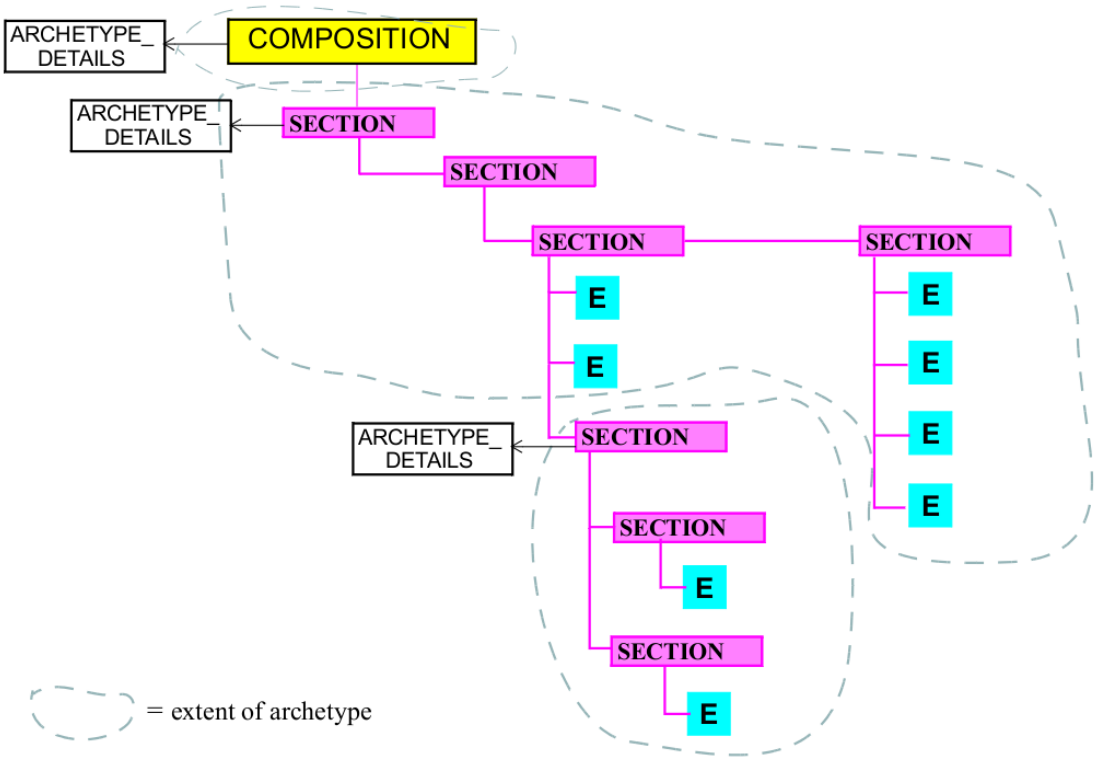

Figure 4.2: Relationship of Compositions, Sections, Entries and Archetypes [25]

their information. Modern relational databases do support a wide range of functions and there are several established products available as open source. But as the name suggests relational databases are based on a relational data model. In contrast, OpenEHR specifies a hierarchical and due to the archetype-concept, highly flexible, data model.

Another technology, which became popular in the past few years are No-SQL databases. Those No-SQL databases are not based on a relational data model. There are three types of No-SQL databases.

- A key/value store - This type of database only holds key-value pairs. Those databases are highly flexible and scalable, but the modeling of relationships between tuples is difficult.
- A document based store - The database only stores documents with meta data attached to it.

- A graph-based database - Graph-based No-SQL databases support a hierarchical data model. All information is stored in a graph, where nodes hold information. Additionally there can be meta data attached to a node.

For the proof of concept, a graph-based No-SQL database has been selected, because it fully supports the hierarchical data model, specified by the OpenEHR standard. Specifically, Apache Jackrabbit [3], an implementation of the JSR-170 [4] and JSR-283 [5] has been selected, because this specification provides extensive versioning features out of the box.

Service Layer

The service layer of the EHR core system holds the business logic. This includes aggregation or composition of data, as well as an input validation.

The service layer has been built using the spring-framework [6]. This framework provides support for aspect oriented programming, as well as inversion of control and dependency injection. This reduces the need of tight coupling inside of the service layer itself.

Furthermore, a fault barrier surrounds the service layer. This fault barrier catches all kinds of exceptions and marshals them into a custom EHR core exception. By using this fault-barrier the EHR core system can satisfy the contract, that it only throws EHR exceptions into the webservice layer.

Webservice layer

The webservice layer has to expose the services provided by the layer via defined interface. In case of the EHR core system, the service have been exposed via SOAP and REST.

First of all the choice for using RESTful services to expose the business functionality was quite obvious. Most of the services identified so far have been resource based, like retrieve an EHR root object with the given id, or retrieve all compositions for a given EHR. This is exactly

[3] http://jackrabbit.apache.org/, last access 2011-01-17 12:39
[4] http://jcp.org/en/jsr/detail?id=170, last access 2011-01-17 12:40
[5] http://jcp.org/en/jsr/detail?id=283, last access 2011-01-17 12:40
[6] http://www.springsource.org/, last access 2011-02-21

what REST has been designed for. The payload of these service is marshaled as XML using xStream [7].

After the first implementation an interoperability issue came up with using .NET to build services depending on the EHR core system. Because the OpenEHR reference implementation is only available in the Java programming language, .NET requires WSDL files to create stubs. Therefore all services exposed via REST have also been exposed via SOAP.

ApacheCXF has been used as a technical framework to expose both, REST and SOAP services. The choice was made in favor of ApacheCXF, because the integration with the spring-framework was good and the documentation of CXF concerning REST and webservices was extensive.

4.3 Exposed interface

The functions exposed by the EHR core system are described in the following sections. They are split up in functions related to the EHR root object, the compositions of an EHR as well as the object reference resolver.

EHR-related functions

The exposed functions which are related to the EHR root object are relatively simple. The EHR core system exposes a method for creating an EHR stub, called createEHR. This method can be used to create a stub object of the EHR root object containing the EHR core system's specific parameters, like the system-ID. Furthermore, the interface provides a method for storing the EHR root object. Therefore a complete EHR root object has to be passed to the method. This function only supports storing of EHRs with an ID, that has not been assigned already. Because of the immutability of the EHR root object it cannot be modified either. Last but not least the interface exposes a retrieval mechanism for finding EHR root objects based on their ID. The ID of the EHR object has to be in the form root::extension [25].

```
/**
```

[7] http://xstream.codehaus.org/, last access 2011-02-07 21:01

```
  * Retrieve an EHR based on its ID
  * @param id identifier in the form root::extension
  * @return an EHR object if found, null otherwise
  */
@WebMethod
EHR getEhr(@WebParam String id);

/**
  * Saves an EHR Object in the repository
  * @param ehr complete EHR object
  * @return an EHR object if saved correctly
  */
@WebMethod
EHR saveEHR(@WebParam EHR ehr);

/**
 * Create a stub EHR object with the correct system-ID of
 * the EHR core system
 * @return EHR root object stub
 */
@WebMethod
EHR createEHR();
```

Listing 4.1: Interface of the EHR core system for the EHR's root object related functions

Composition-related functions

The exposed functions which are related to the storing and retrieval of compositions are quite similar the EHR functions. It provides functionalities for creating, storing and retrieving compositions. But in contrast to the EHR-related functions, it is possible to modify an already stored composition. Although the functionalities for versioning haven't been exposed yet, all compositions are completely versioned by the Jackrabbit framework. Furthermore, each composition is bound to an EHR root object. Therefore the ID of the root object is needed in order to retrieve the correct composition or attach a new composition to the correct EHR. According to

the OpenEHR's information model compositions contain sections and entries (see figure 4.2). Those data-fields are stored in the composition object itself.

```
/**
 * Create an empty composition. If the parentId is set
 * it will be stored as a child of the given composition
 *
 * @param parentId Id of the parnet Composition
 * @return the created Composition
 */
@PUT
@Path("/{parentId}/composition")
String createComposition(@PathParam("parentId") String parentId);

/**
 * Modifies an existing composition
 *
 * @param parentId the id of the EHR-root object
 * @param composition the composition to be stored
 * @return composition the stored composition
 */
@POST
@Path("{parentId}/composition")
String saveComposition(@PathParam("parentId") String parentId, @Multipart(
     value="composition", type="application/xml") String composition);

/**
 * Retrieve a Composition
 *
 * @param parentId the ID of the EHR-root object
 * @param id identifier of the composition
 * @return composition the composition if found
 */
@GET
@Path("/{parentId}/composition/{id}")
```

```
String getComposition(@PathParam("parentId")String parentId, @PathParam("id")
     String id);
```

Listing 4.2: Interface of the EHR core system for the composition related functions

Object reference resolver

Another important concept, which is heavily used in the OpenEHR's reference implementation is the object reference. In order to support interoperability between different OpenEHR implementations, the relationships of different EHR objects are modeled with so called "object references". An object's locatable path is defined by three different attributes:

- Namespace - Namespace to which this identifier belongs in the local system context (and possibly in any other OpenEHR compliant environment) e.g. "terminology", "demographic".
- Type - Name of the class of object to which this identifier type refers, e.g. "PARTY", "PERSON", "GUIDELINE" etc. These class names are from the relevant reference model.
- ID - Identifier of the object

With these three attributes any object within the EHR core system can be uniquely specified.

```
/**
 * Retrieve the object with the given namespace/type/id combination
 *
 * @param namespace namespace of the object
 * @param type type of the object
 * @param id id of the object in to form root::extension
 * @return the object if found marshalled in XML
 */
@GET
@Path("obj/{namespace}/{type}/{id}")
String resolveObjectRef(@PathParam("namespace") String namespace, @PathParam
     ("type") String type, @PathParam("id") String id);
```

Listing 4.3: Interface of the EHR core system for the object resolver

For example when calling the resolveObjectRef function with the parameters Namespace = “ehrCore”, Type=“EHR” and ID=“1234” we would retrieve the EHR root object with the Id 1234 of the ehr core system.

CHAPTER 5

Integration Framework

A stable, reliable and extensibility software architecture is the foundation any enterprise software solution is based on. The architecture of a software system describes its basic design principles as well as its strategies for data integration, function integration or work-flow integration [65]. Especially enterprise systems in the health care context have to integrate several different specialized subsystems by using standardized interfaces.

5.1 Requirements

The requirements to software architecture can be analyzed quite similar to the functional requirements of a software application itself. They arise from functional as well as non-functional requirements, organizational factors as well as social and cultural factors. So the requirements engineering for software architecture includes working with all stakeholders and gathering all their requirements.

In a computerbased information systems in health care or more specifically a hospital environment the most critical requirements are [57]:

- The integration of new subsystems should be as easy and seamless as possible.
- Different subsystems have to be interoperable.

- The system has to comply with current security and privacy standards.
- The communication inside the architecture has to be reliable and persistent.

Those requirements arise from a technical perspective but they also address legal issues.

5.2 Architectural concept

As already mentioned in 2.2, the architectural style we used to satisfy those requirements was a service oriented architecture. By using this style a loose coupling of components is guaranteed which comes along with a high interoperability and easy integration of subsystems. This gives the system the maximum flexibility in terms of integrating new subsystems into the hospital information system.

Furthermore, the whole system is based on a complex one-way-messaging event processor. This concept is based on asynchronous communication, where message distribution is controlled by a rules engine. These components are described in detail in sections 5.2 and 5.3.

The main issues which are addressed with this architecture are

- It has to be possible to integrate all standard or de-facto standard protocols and tools with minimal overhead. The integration of a new protocol/tool should not be more complex than writing a plugin to marshal and unmarshal the requests and responses in XML. Those messages should be enriched with the required meta information and be placed on the HIS' message processing queue.
- Those plugins do not have to know each other and do not have to communicate with each other directly. Instead they should just place events with a additional meta information on the HIS' message processing queue.
- It has to be possible to distribute the plugins on different physical systems, in order to guarantee high scalability.

Messaging infrastructure - Enterprise Service Bus

In the last section, the HIS' message processing queue was mentioned several times, but the technical infrastructure of this messaging system has not been discussed yet. As already stated in the requirements section (see 5.1) all messaging in a hospital information system has to be reliable and persistent. In the Java enterprise world there is a popular standard, called Java Messaging System (JMS), which addresses exactly this problem.

The JMS specification describes two data structures, queues and topics, which are an implementation of a producer-consumer and the publish-subscriber pattern. These data structures can be used in a persistent mode where all messages are stored in a persistent storage, which could be a database as well as a file system. In case of exceptions during the processing of a message, the message is placed on the data structure again, so the communication is persistent as well as reliable.

Because handling multiple queues and topics without a framework isn't too comfortable there are multiple state-of-the-art JMS-frameworks. There are various frameworks available, from simple messaging frameworks, like Apache ActiveMQ [1] over data distribution platforms, like Hazelcast [2] to high-end service oriented infrastructure frameworks, like enterprise service buses [3].

The framework which supports our service oriented architecture the best is an enterprise service bus. ESB's provide a lot of enterprise features, like security, transaction management, business process management as well as naming and directory services. All of those features are based on a reliable service oriented infrastructure. For the practical part the enterprise service bus of JBoss have been chosen, because of its extensive documentation, its license (LGPLv2.1) but most of all because it integrates a lot of established java frameworks, like the drools rule engine [4].

[1] http://activemq.apache.org/, last access 2011-01-27 12:28
[2] http://www.hazelcast.com/, last access 2011-01-27 12:29
[3] http://www.jboss.org/jbossesb, last access 2011-01-27 12:30
[4] http://www.jboss.org/drools, last access 2011-02-07 21:15

Content- and rule based routing of messages

The most integral part in this architecture is the distribution of messages. The main concept of our service oriented architecture is based on reliable and persistent one-way messaging, so the whole communication is carried out asynchronously. Another important design principle is, that the different components deployed on the enterprise service bus do not have to know each other. The most important argument for using asynchronous communication is, that requests can deliver multiple results. In addition, the client does not have to wait for a synchronous result of the request and can carry out other computations without blocking the process.

In order to satisfy the two requirements mentioned before, a plugin called ESB-core has been developed. This plugin is responsible for the distribution of messages based on specific meta information. So the whole system is a complex event-based system, which is controlled by a set of rules.

One message can either be sent to zero, one or more recipients. In order to achieve this, each plugin has to be registered in the ESB-core plugin, with a service category and a service name. So each plugin has to provide a defined interface which can be invoked by the ESB-core plugin in case a message should be delivered to this plugin. Last but not least the rules for distributing messages in the ESB-core plugin have to be added or modified for this plugin. Each message processed by the ESB-core plugin has to contain specific meta information in order to be processed correctly. The most important meta information for messages processed by the ESB-core plugin is the so called event type. This event type specifies the type of the message-event, like “REQUEST_RETRIEVE_EHR”. This is the main property the distribution of messages is based on. Furthermore, each message can contain multiple payloads, which are not modified by the ESB-core plugin.

With this technical infrastructure, the plugins do not need to know each other. They only have to send messages enriched with the required meta information, like the type of the message, to the ESB-core plugin. Consequently they can only receive messages from the ESB-core plugin via an exposed interface. In addition, each message has got an unique id. If a message is a response-message, it’s meta data contains the correlating id of the request message so they can be connected by the consumer.

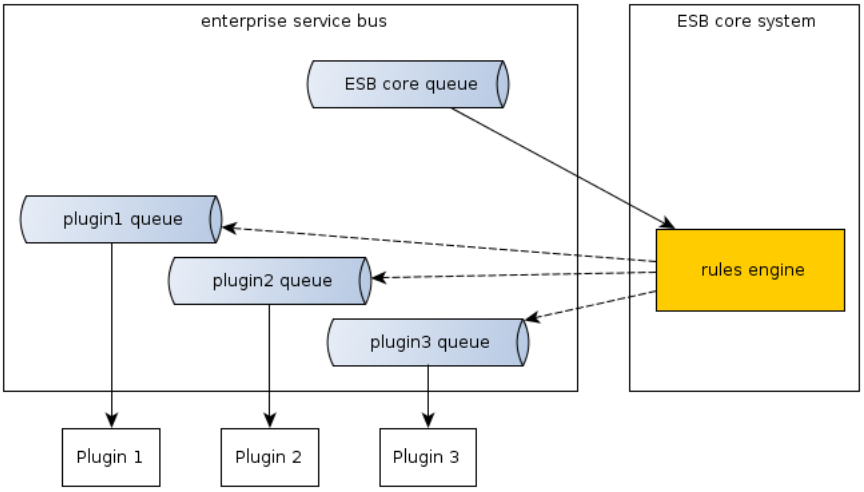

Figure 5.1: ESB-core system architecture

5.3 EHR core system integration and service exposure

As a first step of the proof-of-concept we want to integrate the functions provided by the EHR-core system (see 4) into the hospital information system and expose them over the HIS' architecture. Therefore two separate plugins are needed. Firstly of all we need a plugin, which connects the EHR core system to the HIS', called the med-core plugin. And secondly a plugin is required to expose the functionality by using webservices or RESTful services, called ehr-http plugin. Those two plugins will be described in detail in the following sections.

Med-core plugin

As already mentioned before, a plugin is required which connects the EHR-core system to the HIS' integration architecture. The design of a so called connector is quite simple so are its requirements.

1. Receive a message from the ESB-core system.
2. Unmarshal and parse the message.
3. Invoke the correct functions on the EHR-core system.

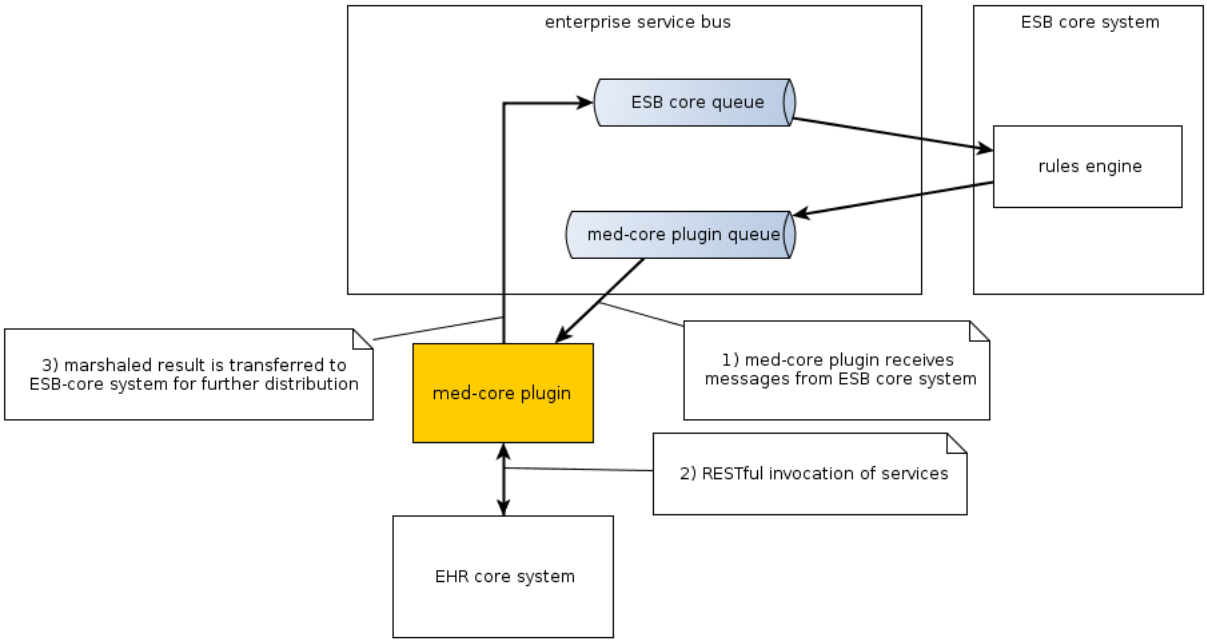

Figure 5.2: Med-core plugin communication

4. Parse and marshal the response of the invocation.
5. Marshal its response payload.
6. Add the necessary meta information to the response message.
7. Place the message on the ESB-core system for further distribution

In order to be able to receive messages from the ESB-core system the med-core plugin has to expose an interface to the HIS. Because we want to comply with a service oriented architecture design, the best way of exposing an interface on an enterprise service bus is to create a message queue and expose its service category and its service name. Additionally, business rules for message distribution have to be added to the ESB-core system. (see listing 5.1)

```
rule "MedCore REQ_RETRIEVE_COMPOSITION"
  when
    m : Message ()
```

```
    p : String(toString == "REQ_RETRIEVE_COMPOSITION")
  then
    destinations.add("MedCore");
end

rule "MedCore RES_RETRIEVE_COMPOSITION"
  when
    m : Message ()
    p : String(toString == "RES_RETRIEVE_COMPOSITION")
  then
    destinations.add("EhrHttpStorage");
end
```

Listing 5.1: ESB-core rules distributing retrieve composition requests and responses

The way the payload is unmarshaled depends on whether it is a plain string or a marshaled XML representation of a java object. If the payload is a String, no unmarshaling has to be done. In case the payload is a XML representation of a Plain Old Java Object (POJO) the java XML framework xStream [5] is used. xStream is a highly configurable and library with excellent performance for serializing and deserializing java objects to XML. Besides the payload which need to be unmarshaled, the type of the event is of high importance. It specifies which method to invoke. This event type is delivered in the meta data of the transferred message. For example the event type "REQ_RETRIEVE_COMPOSITION" indicates that the method for retrieving compositions should be invoked.

In the next step the correct method on the EHR core system has to be invoked. This system exposes SOAP services as well as RESTful services (see 4.2). Because the methods provided by the EHR-core system are resource based the med-core plugin will use the RESTful implementation of the EHR-core services. For this purpose the Apache CXF framework has been used, mainly because it has already been of good use in implementing the EHR core system's webservices.

Once the invocation has been done correctly the result has to be marshaled into XML so it can be transferred. For this purpose we use the xStream library again.

[5] http://xstream.codehaus.org/, last access 2011-01-27 13:32

The last step comprises two dependent tasks. Firstly the message which should be distributed, has to created and secondly it has to be distributed. For the first part the message has to be created and enriched with the necessary meta data. This would be, the correlating id of the request message as well as the correlating response event. For example if the event type of the request message is “REQ_RETRIEVE_COMPOSITION” the corresponding event type would be “RES_RETRIEVE_COMPOSITION”. The REQ prefix indicates a request and the RES prefix indicates a response. Of course the marshaled payload has to be stored in the message as well. Afterwards, the message can be send asynchronously to the ESB-core system for further distribution.

EHR-http plugin

We have specified the med-core connector plugin, which is able to receive messages, invoke methods on the EHR-core system and distribute the response. Now it is necessary to expose those functionalities over the hospital information systems architecture. Due to the asynchronous nature of the architecture this has to be done by in two steps. Firstly we have to expose an interface other services can invoke and secondly we have to publish the result when it is available.

Exposing the functions

Exposing the functions over the functions over the hospital information systems architecture is quite simple. First of all the functions have to be exposed via webservices or RESTful services. The interfaces are similar to the interfaces defined by the EHR-core system (see listing 4.3). If a webservice method is invoked, a message is created which can be passed to the ESB-core system. This message contains the necessary meta information, like the event type. Furthermore, each message has a unique identifier which is stored as a part of the meta information as well. Due to the asynchronous nature of the architecture it is not possible to instantly return a result. To enable to match responses to the corresponding requests, this unique identifier is returned as a result of the invoked webservices. Fetching the result for a specific request can than be done by using this identifier.

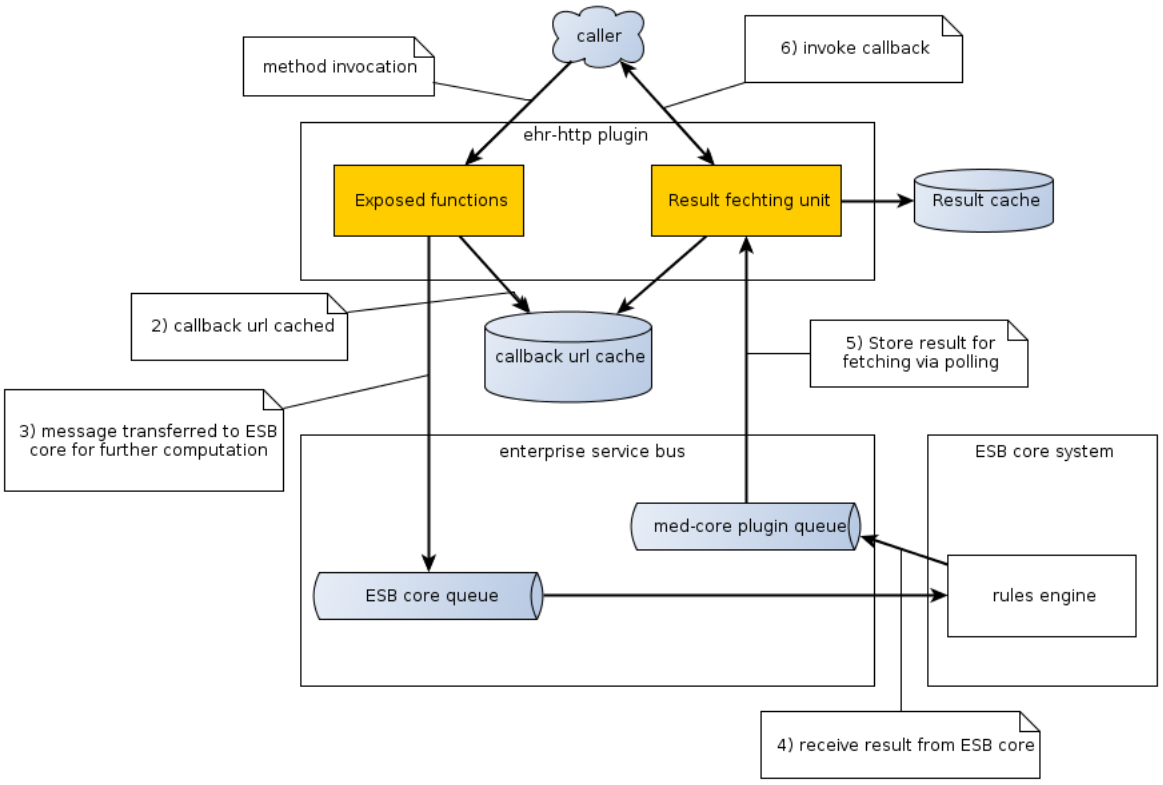

Figure 5.3: ehr-http plugin communication

Retrieving the result

Fetching the result is a key issue when using asynchronous communication patterns. There are two best practices when it comes to returning the result to a caller. The first approach is, that the result is stored in a data structure and can be fetched via polling. This approach is implemented in the J2EE world by using so called Futures. The second approach is, that the caller implements a specified interface and provides a callback URL where the callee can deliver the result once it is computed.

In the practical part of this thesis both approaches have been implemented in order to guarantee the highest flexibility for consumers of the service.

The polling approach is implemented quite straight forward. A webservice method has been implemented which returns results based on the unique identifier returned by the request. A mi-

nor problem is, that the result of a request placed on a java message queue by the ESB-core plugin, which does not allow random access based on specific attributes. For the polling approach it is absolutely necessary to access messages based on their correlation identifier. Therefore all results are fetched from the queue and placed on a distributed hash map immediately. The library used for this hash map is Hazelcast [6], which is a highly scalable open source library completely written in the java programming language.

```
/**
* Fetch the result for a request with the given id
* @param id the id of the correlating request
* @return result in xml
*/
@WebMethod
String getResult(@WebParam String id);
```

Listing 5.2: Interface for polling the result of an invocation of a ehr-http method

The second approach is the callback approach. Because the HIS' architecture is a service oriented architecture, the callback has to be designed in a way it complies to the SoA principles. In order to satisfy this requirement the callback is specified as a webservice callback. The interface which has to be implemented by the caller is listed in listing 5.3. The callback specifies an ID parameter which is the correlating identifier of the request as well as a result parameter which holds the concrete result marshaled in XML. So the caller has to implement the callback interface and host it on a webserver of his choice. Besides that the caller has to specify the URL where the callback should be invoked. So the caller has to specify the callback URL on invocation of the webservice (see listing 5.4). In order to have the callback URL available once the result is available it has to be cached in some kind of distributed data structure. Similar to the caching of the result when using the polling approach, a Hazelcast distributed hash map fits perfectly. The callback URL is stored by using the unique identifier of the request as a key. Once the result is available, the callback URL can be looked up in the map and if specified the callback can be invoked.

```
/**
```

[6] http://www.hazelcast.com/, last access 1011-01-28 20:31

```
* Callback interface for retrieving results of ehr-http requests
* @param id correlating id of the request
* @param result result object marshaled in XML
*/
@WebMethod
void handleCallback(@WebParam String id, @WebParam result);
```

Listing 5.3: Callback interface ehr-http method results

```
/**
* Retrieve the data of a patient with the given ID
* @param id     patients ID
* @param callbackUrl  null if no callback required, complete URI to the
    callbackWebservice
* @return an id which can be given to fetch the data with the getResult(
    String) method.
*/
@WebMethod
String getEhr(@WebParam String id, @WebParam String callbackUrl);
```

Listing 5.4: Callback parameter in ehr-http methods

5.4 Sample Request for Retrieving an EHR

Now the components have been defined, it is interesting how they communicate with each other. In this section a complete work-flow for retrieving an EHR-Object from the EHR core system is described (see figure 5.4).

1. Caller invokes the retrieveEHR webservice method on the ehr-http plugin.
2. The ehr-http plugin publishes a message on the ESB core queue.
3. The ESB core system consumes the message.
4. The rules engine processes the message according to its rules (“MedCore REQ_RETRIEVE_EHR”) and publishes a new message on the med-core plugin queue.

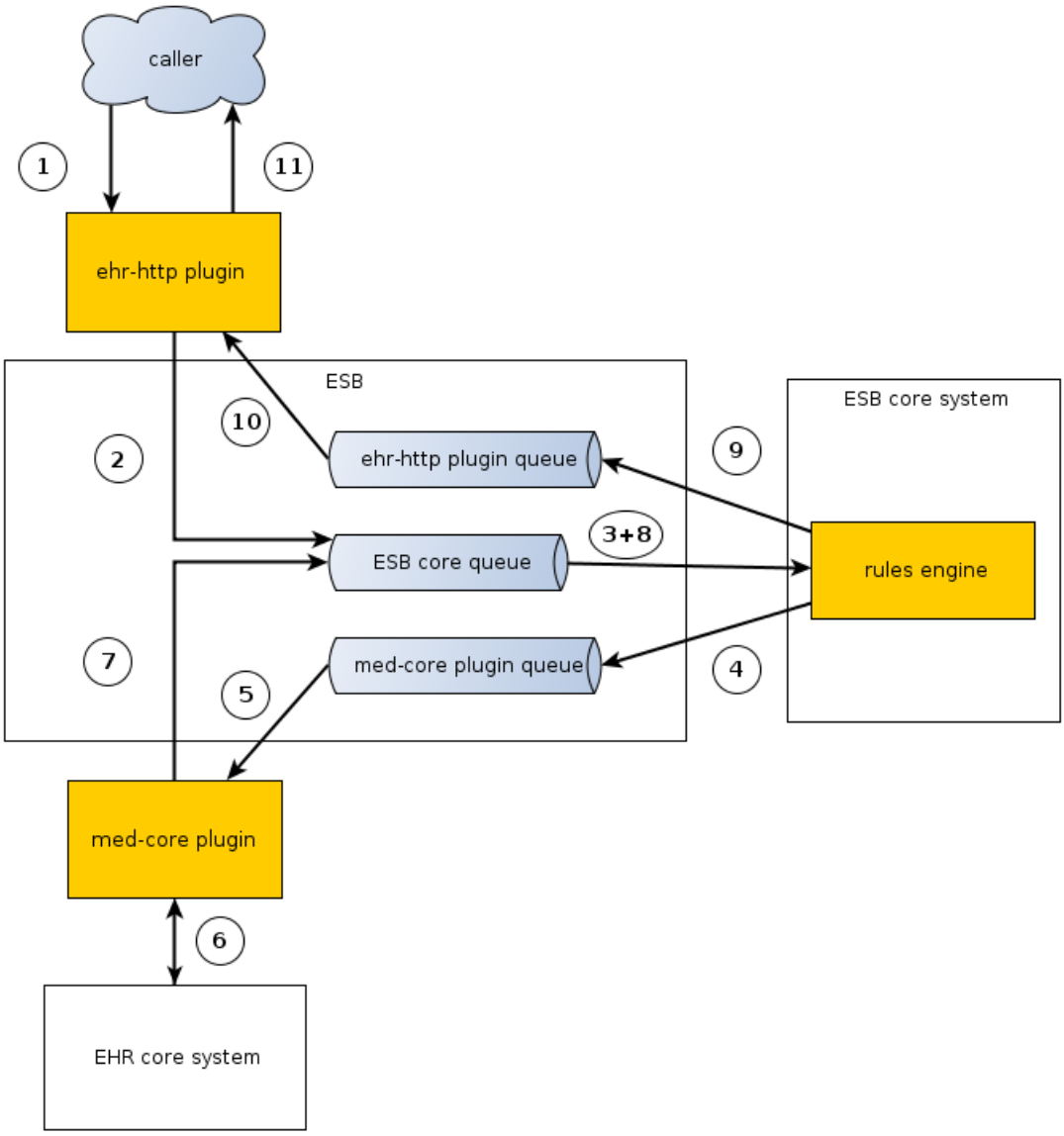

Figure 5.4: Sample request for retrieving an EHR

5. The med-core plugin consumes the message.
6. According to the message, the med-core plugin invokes the retrieveEHR method on the EHR core system.
7. The result of this call is marshaled and published on the ESB core queue.
8. The ESB core system consumes the event.
9. According to the rules defined in the drools engine (“MedCore RES_RETRIEVE_EHR”) a new message is published on the ehr-http plugin queue.
10. The ehr-http plugin consumes the message.
11. The webservice callback on the webserver of the client is invoked.

5.5 Design and Developement of an Integration Framework Prototype

In the course of this thesis an integration framework for a hospital information system has been implemented, to prove that the architectural concept works out. The prototyped components can be grouped into three different categories: EHR-core system, HIS integration framework core components and HIS integration plugins.

EHR-core system

The EHR-core system used to store all EHR-related data. The database schema complies with the OpenEHR information model, only a subset of the whole OpenEHR functionality has been implemented. The identified feature set is sufficient to demonstrate a patient’s admission, his operations and treatments as well as his discharge. A more detailed list of limitations is described in 4.

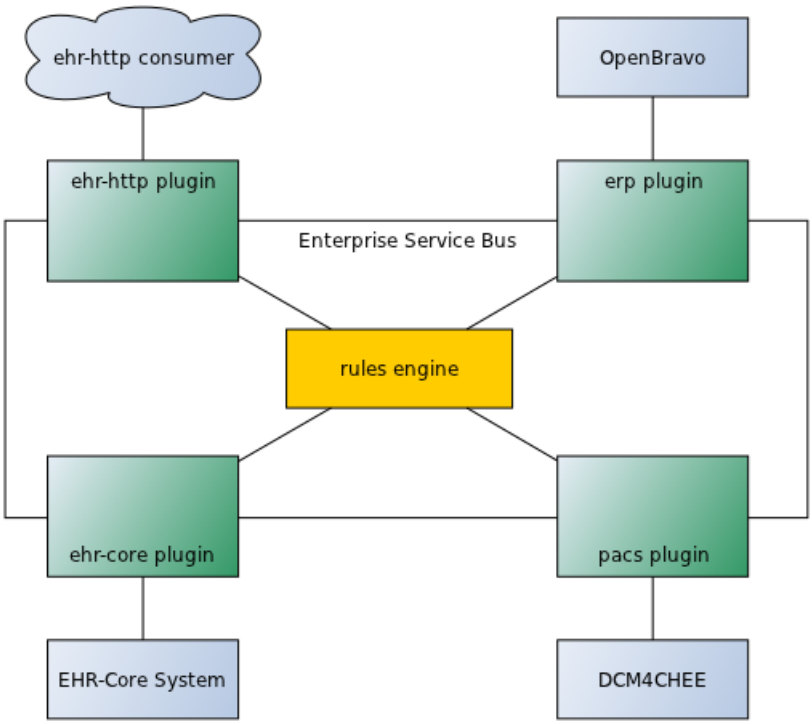

Figure 5.5: Integration Framework Prototype

HIS Integration Framework Core Components

The core components of the HIS integration framework are responsible for providing an infrastructure where integration of plugins can be achieved in a common way. The JBoss enterprise service bus provides this infrastructure. In addition, the rule engine based on drools is responsible for facilitating the communication between different services (see 5.2).

The other core components, the ehr-core-plugin and the ehr-http-plugin are responsible for exposing EHR-related data. The ehr-core-plugin encapsulates the EHR-core-system and provides functions for storing and retrieving EHR-data (see 5.3). The ehr-http-plugin exposes the EHR-related data via webservices to different clients. This plugin provides the same functionality as the ehr-core-plugin (see 5.3). The implementation of clients of the ehr-http-plugin has not been part of the proof of concept.

HIS Of-The-Shelf Integration

The main goal of this thesis is to provide an way to easily integrate different subsystems into a HIS core system. As a proof-of-concept an ERP tool (Openbravo) as well as a PACS (DCM4CHEE) have been integrated. The integration of the ERP tool is limited to updating the stock in a defined warehouse (see 6.3). The integration of the PACS is limited to the retrieval of a patient's data stored (see 7.3.

CHAPTER 6

ERP System Evaluation and Integration

To prove that the architectural concept which has been designed in 5 works out, we want to integrate an enterprise resource planning tool. Therefore we need to evaluate different ERP tools in order to find the one which fits our needs the best in the context of our architecture. To evaluate the ERP tools we use the DESMET feature analysis (see 3.2).

6.1 Requirements of ERP systems

According to the DESMET feature analysis [63] a set of desired features have been described. For every feature a metric has been designed which defines a score. The more of the required or desired functionalities a feature provides the higher is the score. Those scores range from -1 (worst) to 5 (best) for every feature to make them comparable.

In addition, each feature has got an importance which can be mandatory, highly desirable, desirable or nice to have. Mandatory features are must-have features. If a product gets a negative score on a mandatory feature it is eliminated. When all the products are evaluated a final score is calculated to determine the product which fits best. In order to make the different degrees of

Score	Description
-5	Wrong or irritating documentation
0	No documentation
1	Frequently asked questions
2	Up to date documentation (html/pdf) for the latest and each major version
3	Tutorials for common actions (e.g. installation, first steps)
4	Screen casts for common actions (e.g. installation, first steps)
5	Online demo server with sample data

Table 6.1: Evaluation metric for ERP-documentation

importance visible in the final score, mandatory features scores are multiplied by four, highly desirable by 3, desirable by 2 and nice to have features keep their score.

Documentation

Importance: Highly desirable

Because modern enterprise resource planning tools are of high complexity a good documentation is absolutely important. Modern documentations can have various forms [66]. The most common form of documentation, which nearly all software projects provide, is a list of frequently asked questions. Those so called FAQs give answers to common pitfalls. Another form of documentation, normally is the most extensive one, is the traditional written documentation which is provided in a common format (html or pdf). For new users tutorials are good sources of information. They provide information about common actions, for example installation or first steps). Because some tasks are hard to explain with words, screen casts can help users to repeat the tasks without problems. Furthermore, it is very hard to get a first impression of the product just by reading the documentation. Thats why screen casts and an online-demo server can be very useful when it comes down to decide whether to take a closer look at the product.

License

Importance: Mandatory

The license an enterprise resource planning tool is shipped with is of highest interest to

Score	Description
-5	Copyright
1	Strong copyleft (e.g. GPLv3)
3	Weak copyleft (e.g. LGPL, MPL)
5	No copyright/copyleft (e.g. Apache License)

Table 6.2: Evaluation metric for ERP-licenses

our feature analysis [67]. If a ERP tool is shipped with copyright, which requires the authors permission to redistribute or adapt the tool, it is not suitable for our project. A required form of license would be some kind of copyleft or without copyleft. There is a distinction between strong copyleft, like GPLv3 [1] and weak copyleft, like LGPLv3 [2] or the Mozilla Public License [3]. The desired form of license would be a license without copyleft like the Apache License [4].

Integration technologies

Importance: Mandatory

The integration interfaces and technologies an ERP-tool provides are key features in context of our evaluation. We try to build a hospital information system based on a service oriented architecture (SoA). Therefore it would be highly desirable if the chosen tool exposes interfaces with technologies which support a SoA approach. It would also be possible to use shared memory, shared database schema or vendor specific technologies or interfaces, but the preferred way would be exposed webservice (SOAP) [60] interfaces or in the best case CRUD access to all relevant domain objects via RESTful Services [68].

Support

Importance: Desirable

As already mentioned above modern ERP-tools are of high complexity. Therefore it is necessary to have some kind of support where you can phrase questions or problems and have a

[1] http://www.gnu.org/licenses/gpl.html, last access 2011-01-11 22:51
[2] http://www.gnu.org/licenses/lgpl.html, last access 2011-01-11 22:51
[3] http://www.mozilla.org/MPL/MPL-1.1.html, last access 2011-01-11 22:51
[4] http://www.apache.org/licenses/LICENSE-2.0.html, last access 2011-01-11 22:51

Score	Description
-5	Not working properly, makes things worse
0	No integration technologies or integration-interfaces
1	Shared memory or shared database access
2	Vendor specific technologies or interfaces
3	Interfaces for webservices (e.g. SOAP)
5	REST-Interfaces for all the necessary business-objects (including generated wadl or xsd)

Table 6.3: Evaluation metric for ERP-Integration technologies

Score	Description
0	No support at all
1	Open mailing lists
2	User forum
4	Open developer forum
5	Open up to date bug tracker providing workarounds for known bugs

Table 6.4: Evaluation metric for ERP-support

chance of getting a qualified and helpful answer. The basic form of support, which at least most open source projects provide, is an open mailing list where you can email your questions, hope for an answer or search the mailing lists archive for similar problems. Another way of support is a user- or a developer forum, where users try to help each other or developer try to give support [69]. Furthermore, bug tracker can provide a lot of useful information if they are open to the community and provide workarounds for known bugs and problems.

Material management

Importance: Mandatory

In the context of our evaluation we need a basic implementation of a material management system. CRUD functionalities which are accessible via integration technologies mentioned in 6.1 are needed as well as the possibility to customize work-flows for reducing or increasing the stock of products. Furthermore, a build-in restocking mechanism is desired.

Score	Description
-5	No material management
2	Material management which provides simple CRUD functionalities
4	Material management provides building customized work-flows
5	Build in support for restocking

Table 6.5: Evaluation metric for material management in ERP-Tools

Score	Description
-5	Closed source
5	Open source

Table 6.6: Evaluation metric for open source ERP-Tools

Open source

Importance: Mandatory

It is mandatory for our application to have the source code of the ERP-system available. This gives us the possibility to adapt the system to our needs or remove functionality which is not needed at all [70].

6.2 Evaluation of Different ERP Systems

Before the real DESMET feature analysis can be started, a set of possible ERP-tools have to selected. The main criteria was whether the tool was available as open source as well as the maturity of the system. In the process of this screening four tools looked promising. Those four where Openbravo, Compiere, OpenERP and Apache OfBiz. Those four tools then have been evaluated based on the metrics defined in 6.1.

Openbravo

According to the Openbravo website [5] Openbravo is "the leading web-based open source ERP". It supports features like procurement management warehouse and material management or fi-

[5] http://www.openbravo.com, last access 2011-01-11 22:51

Imp.	Feature	Score	Description
M	Open Source	5	Completely open source
M	License	3	Openbravo Public License Version 1.1 – Based on Mozilla Public License (MPL)
M	Integration technologies	5	REST supported via XML and JSON for all domain objects, SOAP-Services for a subset of the domain objects
M	Material management	5	Warehouse management, order fulfillment, work-flow customization
HD	Documentation	5	Screen casts, online demo, tutorials, documentation and FAQs available
D	Support	5	Forums available, custom issue tracker available

Table 6.7: Openbravo feature analysis

nancial management out of the box. The documentation of Openbravo is extensive including screen casts, an online demo server as well as an up to date wiki with tutorials and guides. The support includes a developer forum as well as a custom open issue tracker. Last but not least Openbravo supports CRUD access to all relevant domain objects and furthermore, all database tables by exposed RESTful services. A minor drawback is, that Openbravo does not provide wadl-files which describe the REST services. Instead there is a XSD schema file available describing the data structures which can be consumed by the services. Furthermore, Openbravo is shipped with an Openbravo Public License which is based on the Mozilla Public License (limited copyleft).

Compiere

According to the Compiere website [6] Compiere is “The most modern, adaptable and affordable ERP solution”. Like Openbravo, Compiere supports features like procurement management, warehouse and material management or financial management out of the box. They provide videos as product demos, tutorials and a documentation, but they have no online demo server available. In terms of support Compiere provides a custom issue tracker (called request system)

[6] http://www.compiere.com, last access 2011-01-11 22:51

Importa	Feature	Score	Description
M	Open Source	5	Completely open source
M	License	1	COMPIERE PUBLIC LICENSE based on GPLv3
M	Integration technologies	5	webservices over HTTP/HTTPS (Amazon WS-Style)
M	Material management	5	Warehouse management, order fulfillment, work-flow customization
HD	Documentation	4	Product-demos (videos), tutorials documentation and FAQs available
D	Support	5	Forums available, Custom support request system

Table 6.8: Compiere feature analysis

and a developer forum. Compiere also exposes webservices over HTTP/HTTPS for all business processes. After all a major drawback is, that Compiere comes with the Compiere Public License which is based on the GPLv3, and that Compiere provides professional and enterprise editions which are not freely available as open source.

OpenERP

OpenERP [7] is an open source enterprise resource planning tool which supports operations like CRM, purchase, warehouse management, accounting and human resources out of the box. It is completely open source and the documentation is as extensive as the Openbravo's. This includes online demo servers, screen casts, tutorials as well as standard documentation. The development is coordinated using Launchpad as an issue tracking system and there are developer and user forums available as well. For integration with other systems OpenERP only provides SOAP-services, RESTful services are planned in a future release. OpenERP is shipped with an OpenERP Public License which is based on the AGPLv3, which includes a copyleft.

[7] http://www.openerp.com, last access 2011-01-11 22:51

Importa	Feature	Score	Description
M	Open Source	5	Completely open source
M	License	1	OpenERP Public License based on AGPLv3
M	Integration technologies	3	Service exposed via SOAP, REST support planned in future release
M	Material management	5	Warehouse management, order fulfillment, work-flow customization
HD	Documentation	5	Several screen casts on openerp.tv, online demo, tutorials, documentation, FAQs
D	Support	5	Forums available, Launchpad as bug tracker

Table 6.9: OpenERP feature analysis

Apache OfBiz

Apache OfBiz [8] is an open source enterprise automation software project licensed under the Apache License Version 2.0. Apache OfBiz supports order management, warehouse management, fulfillment, accounting and general work effort management out of the box. The documentation includes an online demo server, extensive screen casts hosted on youtube [9], tutorials and standard documentation. A developer forum is available and so is an issue tracking system (JIRA). Apache OfBiz supports REST only via third party frameworks but provides SOAP endpoints for all business processes.

Final Scores and Decision

As mentioned in 6.1 the final scores are computed as the product of the scored point of the functionality multiplied by the importance-factor. Table 6.11 shows the final scores of each evaluated ERP system compared to the others.

Overall Apache OfBiz and Openbravo did get the highest score. They were tied in mandatory points (72/80) and did get the full score on highly desirable (15/15) and desirable (10/10) features. Apache OfBiz did lack 8 points on the integration technology metric because they only support SOAP services and on the other hand, Openbravo lost 8 points because the license

[8] http://ofbiz.apache.org, last access 2011-01-11 22:51
[9] http://www.youtube.com/ofBiz, last access 2011-02-07 12:43

Importa	Feature	Score	Description
M	Open Source	5	Completely open source
M	License	5	Apache License
M	Integration technologies	3	REST only possible via a third party framework like apache wink, Service exposed via SOAP
M	Material management	5	Warehouse management, order fulfillment, work-flow customization
HD	Documentation	5	Online Demo, screen casts via youtube, tutorials, documentation, FAQs
D	Support	5	Forum available, JIRA as bug tracker

Table 6.10: Apache OfBiz feature analysis

Importa	Feature	Openbravo	Compiere	OpenERP	Apache OfBiz
M	Open Source	20	20	20	20
M	License	12	4	4	20
M	Integration technologies	20	20	12	12
M	Material management	20	20	20	20
HD	Documentation	15	12	15	15
D	Support	10	10	10	10
	Mandatory Score	72	64	56	72
	Highly Desirable Score	15	12	15	15
	Desirable Score	10	10	10	10
	Final Score	97	86	81	97

Table 6.11: Final score of feature analysis

(MPL) it is released with.

The decision was made in favor of Openbravo, because concept of their RESTful service implementation looks promising. Furthermore, the documentation in Openbravo's wiki seems to be very good when it comes to integrating Openbravo into other systems.

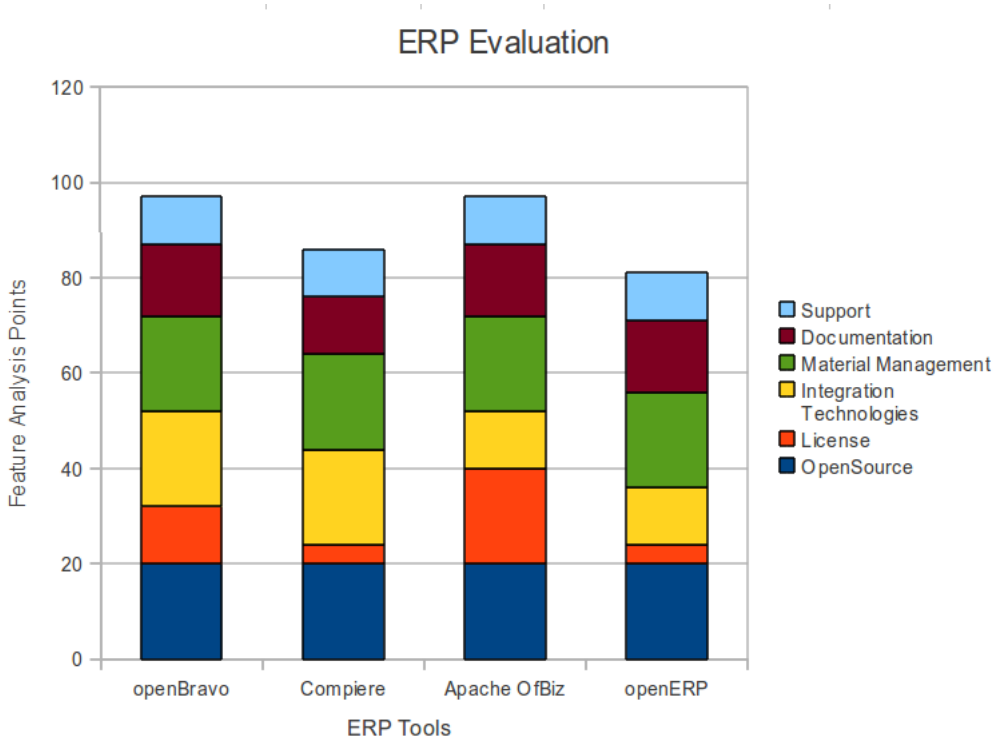

Figure 6.1: Final score of feature analysis

6.3 ERP Integration

After the evaluation is done, the ERP system which got the best score has to be integrated into the hospital information system. As a proof of concept only one use case will be implemented. For this one use case the material management is chosen, because this is one of the most important features an ERP system has to provide in a hospital environment. To sufficiently provide those features a few preconditions have to be set:

- A user based authentication has to be provided
- A material warehouse for medical supplies has to be defined in the ERP system
- An item with name “cotton bud” has to be stored in this warehouse a specific quantity.

The requirements for this use case are rather simple. If we do a physical examination, which is technically speaking storing a physical examination composition, we want to reduce the amount of cotton buds available in this hospital by 2.

Sample Openbravo Setup

First of all Openbravo has to be set up correctly in order to satisfy all the preconditions required for the use case. The initial setup can be done in various ways. Openbravo is available as a traditional installation (e.g. available over the apt repositories) or as deployable package for the amazon EC2 cloud. For this work the traditional installation has been selected.

In the Openbravo default configuration a set of default users, companies, warehouses and items is available. For the sake of simplicity a new item, called “cotton bud” is created in an existing warehouse. Afterwards a load of cotton buds is stored in the warehouse. At last a user with the permissions to reduce the quantity of cotton buds is created. With this setup all preconditions, specified in 6.3, are satisfied.

ERP-ESB Plugin

As described in 5.2 the integration of different subsystems is done via implementing plugins which are deployed on the HIS’ enterprise service bus. The plugin is named ERP-ESB because it integrates the ERP system in the enterprise service bus. The ERP-ESB’s logic is quite simple. It has to receive messages and parses them to find out whether it should trigger an ERP process or not. If the quantity of a medical supply should be reduced, it retrieves the actual quantity from the ERP system and updates the quantity via RESTful services.

One unanswered question question is, how the plugin receives the correct messages. This is done via the content- and rule based routing which is described in 5.2. When a specific event occurs (RES_MODIFY_COMPOSITION the response message from modifying a composition) the ERP integration is invoked automatically. The rule itself complete rule can be seen in the listing 6.1

```
rule "MedCore RES_MODIFY_COMPOSITION"
  when
```

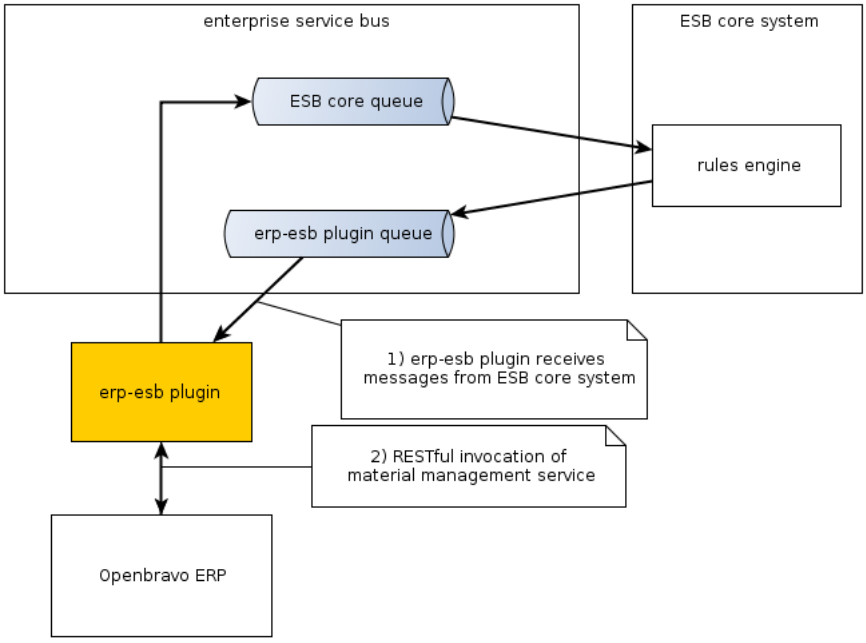

Figure 6.2: erp-esb plugin communication

```
    m : Message ()
    p : String(toString == "RES_MODIFY_COMPOSITION")
  then
    destinations.add("EhrHttpStorage");
    destinations.add("ERPService");
end
```

Listing 6.1: ERP Integration Rule

CHAPTER 7

Picture Archiving and Communication Systems Evaluation and Integration

As a further proof of concept, we want to integrate a PACS into the hospital information system. Therefore we need to evaluate different PACS in order to find the one which fits our needs the best in the context of our architecture. Again we use DESMET feature analysis 3.2 for evaluating different systems. It should be mentioned that this evaluation does not evaluate the clinical use of PACS, it just evaluates the integration functionalities it provides.

7.1 Requirements of PACS

According to the DESMET feature analysis [63] a set of desired features have been described. For every feature a metric has been designed which defines a score. The more of the required or desired functionalities a feature provides the higher is the score. Those scores range from -1 (worst) to 5 (best) for every feature to make them comparable.

In addition, each feature has got an importance which can be mandatory, highly desirable, desirable or nice to have. Mandatory features are must-have features. If a product gets a negative score on a mandatory feature it is eliminated. When all the products are evaluated a final score is calculated to determine the product which fits best. In order to make the different degrees of

Score	Description
-5	No DICOM Integration
1	DICOM Storage
3	DICOM Query/Retrieve
5	DICOM Web Access to DICOM Persistent Objects (WADO)

Table 7.1: Evaluation metric for PACS DICOM Integration

importance visible in the final score, mandatory features scores are multiplied by four, highly desirable by 3, desirable by 2 and nice to have features keep their score.

DICOM Integration

Importance: Mandatory

The most important functionality the PACS has to provide is a proper implementation of DICOM standards. The use case the PACS should be used for requires DICOM storage and a mechanism to retrieve data over a specified DICOM interface. This problem is addressed by two different DICOM standards, the DICOM Query/Retrieve and the Web Access to DICOM Persistent Objects (WADO). Later would be preferable because of the service oriented nature of the reference architecture of the hospital information system itself.

License

Importance: Mandatory

The license a PACS is shipped with is of highest interest to our feature analysis [67]. If a PACS is shipped with copyright, which requires the authors permission to redistribute or adapt the tool, it is not suitable for our project. A required form of license would be some kind of copyleft or without copyleft. There is a distinction between strong copyleft, like GPLv3 [1] and weak copyleft, like LGPLv3 [2] or the Mozilla Public License [3]. The desired form of license would be a license without copyleft like the Apache License [4].

[1] http://www.gnu.org/licenses/gpl.html, last access 2011-01-11 22:51
[2] http://www.gnu.org/licenses/lgpl.html, last access 2011-01-11 22:51
[3] http://www.mozilla.org/MPL/MPL-1.1.html, last access 2011-01-11 22:51
[4] http://www.apache.org/licenses/LICENSE-2.0.html, last access 2011-01-11 22:51

Score	Description
-1	Copyright
1	Strong copyleft (e.g. GPLv3)
3	Weak copyleft (e.g. LGPL, MPL)
5	No copyright/copyleft (e.g. Apache License, BSD, MIT)

Table 7.2: Evaluation metric for PACS-licenses

Score	Description
-1	Closed source
5	Open source

Table 7.3: Evaluation metric for open source PACS-Tools

Score	Description
0	No IHE Profiles used
3	Several IHE Profiles used
5	Design based on IHE Profiles

Table 7.4: Evaluation metric for PACS integration of IHE Profiles

Open source

Importance: Mandatory

It is mandatory for our application to have the source code of the PACS available. This gives the possibility to adapt the system to our needs or remove functionality which is not needed at all.

IHE Profiles

Importance: Highly Desirable

As already mentioned in 1.4, IHE-Profiles are an important concept when it comes to integrating different systems in a health care context. Therefore it is highly desirable for a PACS to support IHE Profiles or have a design based on IHE-Profiles to enable the usage of those when integrating different systems.

Score	Description
-1	Wrong or irritating documentation
0	No documentation
1	Frequently asked questions
2	Up to date documentation (html/pdf) for the latest and each major version
3	Tutorials for common actions (e.g. installation, first steps)
4	Screen casts for common actions (e.g. installation, first steps)
5	Online demo server with sample data

Table 7.5: Evaluation metric for PACS-documentation

Documentation

Importance: Highly desirable

Because modern picture archiving and communication systems are of high complexity a good documentation is absolutely important. Modern documentations can have various forms [66]. The most common form of documentation, which nearly all software projects provide, is a list of frequently asked questions. Those so called FAQs give answers to common pitfalls. Another form of documentation, which normally is the most extensive form of documentation, is the traditional written documentation which is provided in a common format (html or pdf). For new users tutorials are good sources of information, which provide information about common actions (for example installation or first steps). Because some tasks are hard to explain with words, screen casts can help users to repeat the tasks without problems. Furthermore, it is very hard to get a first impression of the product just be reading the documentation. Thats why screen casts and a online-demo server can be very useful when it comes down to decide whether to take a closer look onto the product.

Extensibility

Importance: Highly Desirable

The most fundamental objects in the DICOM standard are service classes and information objects. The combination of these two units is a functional unit of DICOM. These units are called service-object-pair classes (SOPs). The communication between DICOM-compatible devices

Score	Description
-5	New SOPs cannot be added
1	Modification of source code is always required
3	Modification of source code is sometimes required
5	New SOPs can be added at runtime

Table 7.6: Evaluation metric for PACS extensibility regarding adding new SOPs

Score	Description
1	One operating system supported
2	Two operating systems supported
3	Three operating systems supported
5	Multi-platform (more than 3)

Table 7.7: Evaluation metric for PACS regarding supported operating systems

involves exchanging SOP instances by using DICOM messages, which is the communication version of a SOP class.

Therefore it is important for a PACS to support as many service-object-pair classes as possible. Because new SOPs can be developed regularly it is crucial to have the possibility to add those new SOPs without doing invasive changes.

Operating System

Importance: Highly Desirable

Modern PACS should be available on as many platforms as possible. This gives the user the chance to host the PACS of his choice on the operating system of his choice.

Ongoing Development

Importance: Highly Desirable

A vital community and ongoing development and maintenance are signs of a stable product. This metric is of high importance, because it should prevent a so-called “dead” project to be chosen. The most preferable version would be a product with fixed release cycles, but for the sake of simplicity we will evaluate just the latest release date and the latest commit.

Score	Description
-5	No Commit in the last 365 days
0	Last release older than 730 days
2	Last release older than 365 days
5	Last release older than 182 days

Table 7.8: Evaluation metric for PACS ongoing development

Score	Description
0	No support at all
1	Open mailing lists
2	User forum
4	Open developer forum
5	Open up to date bug tracker providing workarounds for known bugs

Table 7.9: Evaluation metric for PACS-support

Support

Importance: Desirable

As already mentioned above modern PACS-tools are of high complexity. Therefore it is necessary to have some kind of support where you can phrase questions or problems and have a chance of getting a qualified and helpful answer. The basic form of support, which at least most open source projects provide, is an open mailing list where you can email your questions, hope for an answer or search the mailing lists archive for similar problems. Another way of support is a user- or a developer forum, where users try to help each other or developer try to give support [69]. Furthermore, bug tracker can provide a lot of useful information if they are open to the community and provide workarounds for known bugs and problems.

7.2 Evaluation of Different PACS

As already discussed in 3.2 a set of possible PACS candidates has to be selected. The main criteria in this search was, whether the tool is an open source tool and implements any DICOM standards. Furthermore, there has already been an evaluation of open source PACS, which

was done by the University of Leipzig [71]. In this article three open source DICOM frameworks have been evaluated. According to the screening those three Frameworks seem to be the most mature open source DICOM frameworks, which are currently available. Those three are DCMTK, DCM4CHEE and CONQUEST, which have been evaluated according to the metrics defined in 7.1.

DCMTK

The DICOM Toolkit [5] (DCMTK) is a collection of libraries and applications implementing large parts of the DICOM standard. It has been used at numerous DICOM demonstrations to provide central, vendor-independent image storage and work-list servers and is used by hospitals and companies all over the world.

The DCMTK is actively developed in the C++ programming language. It is completely open source and licensed under BSD-license. It provides fundamental DICOM functionalities (storage and query/retrieve). DCMTK is available on various platforms and provides a good documentation and well documented source code. In addition, there is a support forum and a wikipage with well known issues or limitations and workarounds.

DCM4CHEE

The DCM4CHE [6] is a collection of open source applications and utilities for the health care enterprise. The DCM4CHEE is an enterprise image manager and image archive (according to IHE). It provides HL-7 and DICOM interfaces which are important for storage, retrieval and work-flow to a health care environment. Furthermore, it provides a web-based user interface for administrating the server.

The DCM4CHEE is developed in the Java programming language and is shipped with a JBoss application server. By taking advantage of the JBoss features (JMS, EJBs, JMX, etc.) and a clean code base, the setup should be familiar for any java developer. As well as the DCMTK, DCM4CHEE is completely open source and released under a triple license (MPL

[5] http://dicom.offis.de/dcmtk.php.en, last access 2011-01-07 11:23
[6] http://www.dcm4che.org/, last access 2011-01-01 11:40

Imp.	Feature	Score	Description
M	Open Source	5	Completely open source
M	License	5	Based on a BSD license (no copyleft)
M	DICOM Integration	3	DICOM Storage,Query/Retrieve Implemented
M	Operating Systems	5	Linux, Solaris, HP-UX, IRIX, FreeBSD, OpenBSD and MacOS X.
HD	IHE Profiles	0	No IHE Profiles
HD	Documentation	3	FAQs, Online Documentation and How-tos available
HD	Extensibility	1	source code modification always required when adding new SOPs
HD	Ongoing Development	5	Last stable release 2011-01-06
D	Support	5	Forums, Wikipage with known issues and workarounds

Table 7.10: DCMTK feature analysis

1.1, GPLv2 and LGPLv2). The DCM4CHEE implements the complete DICOM standard and provides storage, query/retrieve as well as the DICOM web access (WADO). It has a design based on IHE profiles and assumes the role of several IHE actors, to provide a high level of interoperability.

CONQUEST

The CONQUEST [7] DICOM software is a fully featured DICOM server, which is developed in a cooperation of various universities and institutes. It provides a web based user interface as well as simple DICOM operations (query/retrieve, move and modality work-lists).

CONQUEST is completely open source and is made available with a public domain license. It does not implement any IHE profiles and the documentation is sufficient, but not extensive. In contrast a support forum is available where developers are contributing.

[7] http://www.xs4all.nl/ ingenium/dicom.html, last access 2011-01-07 11:48

Imp.	Feature	Score	Description
M	Open Source	5	Completely open source
M	License	3	released under a triple license of Mozilla Public License version 1.1 (MPL), the GNU General Public License version 2 or later (GPL), or the GNU Lesser General Public License version 2.1 or later (LGPL)
M	DICOM Integration	5	DICOM Storage,Query/Retrieve, WADO Implemented
M	Operating Systems	5	Multi-platform (based on Java)
HD	IHE Profiles	5	Based on IHE Profiles
HD	Documentation	5	FAQs, Online Documentation and How-tos available, Online Demo Available
HD	Extensibility	3	source code modification sometimes required when adding new SOPs
HD	Ongoing Development	5	Last stable release 2010-08-27
D	Support	5	Forums, and JIRA as bug tracker with workarounds for known issues available

Table 7.11: DCM4CHEE feature analysis

Imp.	Feature	Score	Description
M	Open Source	5	Completely open source
M	License	5	Public Domain
M	DICOM Integration	3	DICOM Storage,Query/Retrieve Implemented
M	Operating Systems	2	Unix,Windows
HD	IHE Profiles	0	No IHE Profiles
HD	Documentation	2	Online Documentation
HD	Extensibility	1	source code modification required
HD	Ongoing Development	2	Last stable release 2009-09-29
D	Support	4	Forums with developers contributing

Table 7.12: CONQUEST feature analysis

Imp.	Feature	DCMTK	DCM4CHEE	CONQUEST
M	Open Source	20	20	20
M	License	20	12	20
M	DICOM Integration	12	20	12
M	Operating Systems	20	20	8
HD	IHE Profiles	0	15	0
HD	Documentation	12	15	6
HD	Extensibility	3	9	3
HD	Ongoing Development	15	15	6
D	Support	10	10	8
	Mandatory Score	72	72	60
	Highly Desirable Score	30	54	15
	Desirable Score	10	10	8
	Final Score	112	136	83

Table 7.13: Final score of the PACS feature analysis

Final Decision

As mentioned in 3.2 the final scores are computed as a product of the scored point of the functionality multiplied by the importance-factor. Table 7.13 and figure 7.1 show the final scores of each evaluated PACS compared to the others.

In this evaluation the DCM4CHEE PACS was the clear winner. The tool scored the most points overall (136 out of 151 possible) and in addition, it did get the best score in each importance category. This evaluation leads us to a similar result as it was described in the open source DICOM framework evaluation done by the University of Leipzig in 2006 [71]. Given the result, the DCM4CHEE PACS will be used to be integrated into the hospital information systems architecture.

7.3 PACS Integration

As a further proof of concept for the architecture designed in 5.2 a PACS is integrated into the hospital information system. Given the result of the software evaluation the DCM4CHEE PACS is used. As a sample use case the retrieval of patient data from a PACS is selected, because it

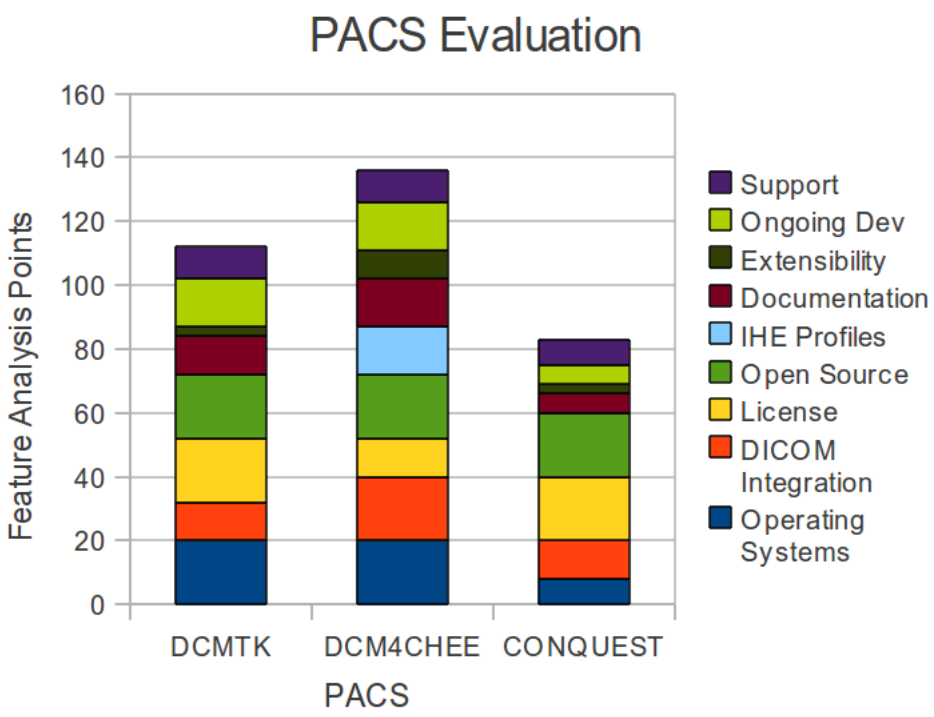

Figure 7.1: Final score of PACS feature analysis

can extend the functionality of the retrieval of patient data from the OpenEHR repository. In order to guarantee a seamless integration of the PACS into the HIS, the PACS has to support the DICOM c-find functionality.

Sample DCM4CHEE Setup

The wiki of DCM4CHEE describes various ways of installing a stand-alone version of the DCM4CHEE PACS [8]. In order to avoid problems with the database configuration the distribution using HSQLDB [9] as a database backend is selected. HSQLDB is pure java in-memory database which does not require a stand-alone server. For the proof of concept the binary distribution and the contained scripts had been used. Following the Step-by-Step installation guide on the DCM4CHEE wiki a JBoss application server is configured and the DCM4CHEE is de-

[8] http://www.dcm4che.org/confluence/display/ee2/Installation, last access 2011-01-07 18:29
[9] http://hsqldb.org/, last access: 2011-01-07 18:31

ployed on this server. Because the hospital information system itself is running on a JBoss AS, the server hosting the PACS has to use a different set of port bindings in order to avoid port conflicts. Therefore a new JBoss binding file [10] is used.

After completing these steps, the PACS web application as well as the DICOM interfaces are hosted on this application server. To satisfy all required preconditions a patient with a specific patient-ID has to be created using the web interface.

PACS-ESB Plugin

Similar to the ERP integration, a plugin for the enterprise service bus has to be implemented to allow communication between the hospital information system and the PACS itself. This plugin has to provide the functionality to consume messages delivered by the ESB-core. Those messages contain IDs of the patients where PACS data is requested. The plugin should retrieve this data from PACS.

The PACS plugin is triggered by the ESB-core where specific rules are defined for retrieving patient data. The retrieval of patient has already been implemented in 5.3, but for the integration of the PACS these functions had to extended. There had been changed to the ehr-http plugin as well as to the esb-core.

First of all the signature of the getEHR webservice method changed. A new parameter has been introduced, which specifies the retrieval of the patient data. Now the data can be retrieved from the either the PACS, the EHR core system or from both storages. The parameter is stored as a header-parameter in the transferred ESB-message. This did also lead to changes in the getResult webservice method (see listing 7.1). Before integrating the PACS the HIS got at most one response for each request. Now there can be either one or two responses which have to be stored by the ehr-http plugin until they are fetched. In the process, the Hazelcast map which is used for storing these results changed as well.

```
/**
* Retrieve the data of a patient with the given ID
* @param id     patients ID
```

[10] http://docs.jboss.org/jbossas/docs/Server_Configuration_Guide/4/html/Services_Binding_Management-The_Sample_Bindings_File.html, last access: 2011-01-07 18:32

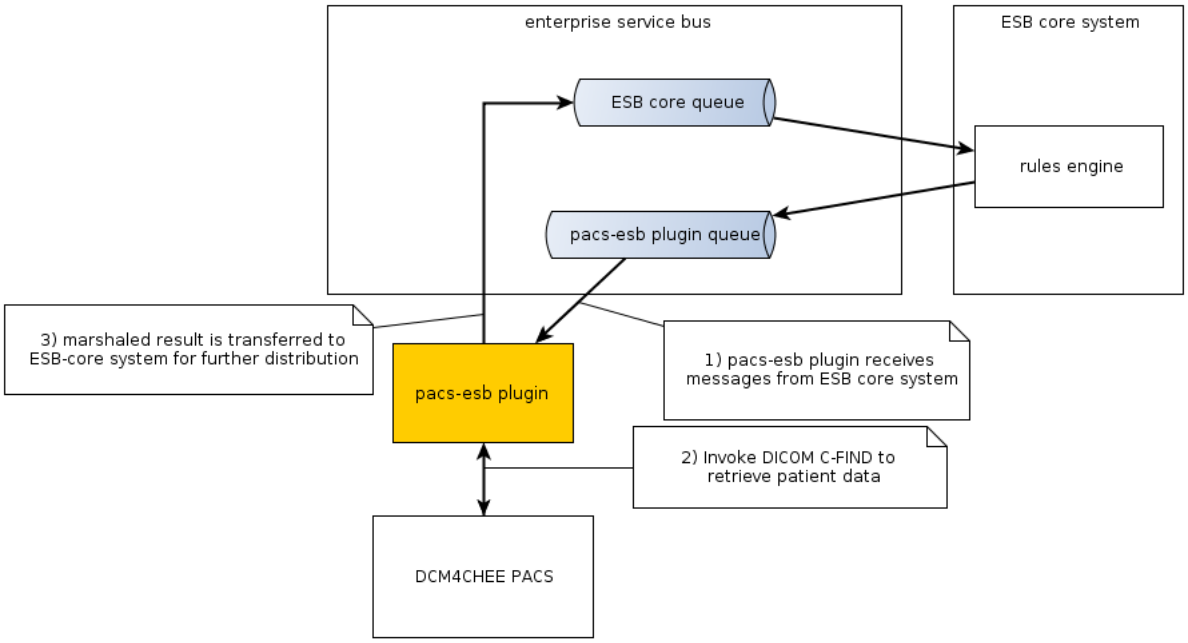

Figure 7.2: pacs-esb plugin communication

```
* @param callbackUrl  null if no callback required,
*   complete URI to the callbackWebservice
* @param retrieveEhrType   type of the source where the patient's data
*   should be fetched. Can be either PACS, MEDCORE or MEDCORE_WITH_PACS
* @return an id which can be given to fetch the data with the
*   getResult(String) method.
*/
@WebMethod
String getEhr(@WebParam String id, @WebParam String callbackUrl, @WebParam
     RetrieveEhrType retrieveEhrType);

/**
* Fetch the result for a request with the given id
* @param id the id of the correlating request
* @return a list of result in xml
*/
@WebMethod
```

```
List<String> getResult(@WebParam String id);
```

Listing 7.1: Modified webservice methods in the ehr-http plugin

The changes to the esb-core are limited to a minor adaption in the file which describes the rules for the message distribution. The rule for retrieving a patients health record is now split into three rules corresponding to the storage where the data should be fetched. Those rules are based on the *event-key* as well as the *RetrieveEhrType* header parameter (see listing 7.2).

```
rule "MedCore REQ_RETRIEVE_EHR_MEDCORE_WITH_PACS"
  when
    m  : Message ()
    p  : String(toString == "REQ_RETRIEVE_EHR")
    rt : String(toString == "MEDCORE_WITH_PACS")
  then
    destinations.add("PacsService");
    destinations.add("MedCore");
end

rule "MedCore REQ_RETRIEVE_EHR_PACS"
  when
    m  : Message ()
    p  : String(toString == "REQ_RETRIEVE_EHR")
    rt : String(toString == "PACS")
  then
    destinations.add("PacsService");
end

rule "MedCore REQ_RETRIEVE_EHR_MEDCORE"
  when
    m  : Message ()
    p  : String(toString == "REQ_RETRIEVE_EHR")
    rt : String(toString == "MEDCORE")
  then
    destinations.add("MedCore");
```

```
end
```

Listing 7.2: ESB-core rules for triggering the PACS plugin

When the plugin is triggered, it uses an open source DICOM framework [11] to connect to the PACS. After the connection is established the plugin uses a DICOM c-find to retrieve all relevant data of a patient (id, name, birth date, sex). This data is marshaled into an XML object and put onto the esb-core's queue, where it is forwarded to the ehr-http plugin.

[11] http://www.dcm4che.org/confluence/display/d2/dcm4che2+DICOM+Toolkit, last access 2011-01-07 19:33

End

CHAPTER 8

Results

Especially in the context of hospital information systems it is common to integrate several of the shelf products like ERP systems, PACS, decision support systems or LIS. Those systems expose certain interfaces which can rarely by changed and therefore, it is essential to provide a common infrastructure to enable communication between different systems. Because of the defined interfaces of such tools it is unavoidable to create a connector component to wrap the services of the system which should be integrated. In the course of this thesis a prototype for a hospital information system based on a service oriented architecture has been developed. The core system makes use of an enterprise service bus for implementing a complex event based messaging infrastructure. The distribution of messages is then handled by a drools rule engine. Java message queues are used in order to guarantee reliable and persistent messaging.

Beside the HIS' core information system an EHR core system was designed and developed, which is based on the OpenEHR standard. This system made use of the OpenEHR reference implementation and exposed RESTful as well as SOAP services for creating, modifying and retrieving patient data. For storing that information a PostgreSQL database was used in combination with the Apache Jackrabbit framework.

The services for creating, modifying and retrieving electronic health records and OpenEHR compositions are exposed by a separate module. This component only uses the HIS' core system to publish and retrieve messages. The HIS' core system is responsible for publishing messages

to another plugin, which acts as a connector for the EHR core system. This plugin invoks the correct methods on the EHR core system and forwards the results onto HIS' core system. The HIS' core system subsequently distributes the results to the plugin which exposed the webservices in the first hand.

As a proof-of-concept for the integration framework which has been implemented, a picture archiving and communication system (PACS) as well as a enterprise resource planning (ERP) tool had to be integrated. In order to find the best suitable PACS and ERP systems, all of them had to be evaluated using the DESMET software evaluation method. Therefore, several requirements for these systems have been defined and each of the candidates has been evaluated by these criteria. Afterwards, the best system according to the DESMET score was chosen to be integrated into the HIS' architecture. For each system one use case was implemented. The PACS is used to retrieve PACS-specfici patient data and the ERP system keeps the material warehouse up-to-date.

8.1 Conclusion

The motivation for this thesis was roused by two interesting research questions. Firstly, how chould a hospital information system be designed in order to guarantee highest flexibility in integrating different subsystems and secondly, can the OpenEHR standard with its reference implementation be used in practice?

In current enterprise software engineering projects there is a strong trend towards the use of service oriented architectures. They enforce loose coupling of components and the use of standardized interfaces and therefore support simple integration of new services. Those properties of a service oriented architecture fit perfectly into the context healthcare. There are several open-source frameworks which support service-oriented architectures and enterprise application integration. For the practical part of this thesis the JBoss Enterprise Service Bus had been used as it provides lot of enterprise features, like security, transaction management, business process management as well as naming and directory services which can be of use for further features.

As has been mentioned before a PACS and an ERP system had to be integrated into the

HIS in order to prove that the integration of services could easily be possible. When integrating the PACS system the changes in the HIS core system were only localized in the file describing the drools rules for the distribution of messages. The development of the connector plugin was quite straightforward as well, because the plugin only had to invoke an interface which exposes functionality described in the DICOM standard (DICOM C-FIND). The lion's share of the work was to install and configure the standalone PACS. Quite the same applies to the integration of the ERP system.

All in all, the integration of both, the PACS as well as the ERP system into the HIS, was quite uncomplicated and straightforward and therefore service-oriented architectures seem to be promising in the context of hospital information systems.

The second question was whether OpenEHR could be used in practice. Of course, the significance of the practical work is limited to the use cases which have been implemented (creating, modifying and retrieving electronic health records and OpenEHR compositions), but for those use cases the OpenEHR standard and in particular the OpenEHR reference implementation could be used extremely well. The documentation concerning the OpenEHR architecture as well as the OpenEHR information model was extensive and easily understandable. Furthermore, the OpenEHR reference implementation is based on state-of-the-art software development and deployment frameworks, like maven [1], and its documentation and coding style is of high quality.

In retrospect, the concept of the integration architecture based on the OpenEHR standard proved to work well for integrating different information systems.

8.2 Further work

An enhancement of high value would be the development of a graphical user interface which would have to be dynamically rendered by the rules defined by OpenEHR templates and archetypes. There could be several different types of the user interface adapted to the context of its use. For example a medical view for physicians, supporting the hospital staff, a patient view, where patients can retrieve their data, or a view for teaching, where students can do diagnosis based on

[1] http://www.maven.org, last access 2011-02-04 14:12

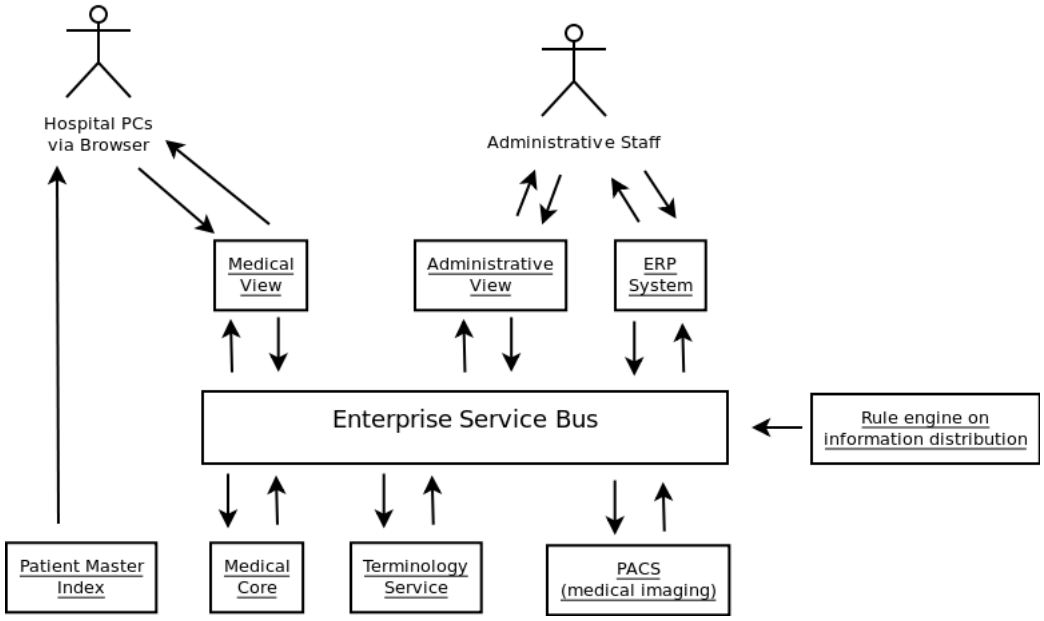

Figure 8.1: Possible further topics

specific facts. In this context the integration of a patient master index would be valuable as well.

In the course of this thesis a common infrastructure for a hospital information system has been designed. A lot of components could be integrated using this infrastructure in order to carry out specific tasks. For instance decision support systems could be connected to the EHR core system in order to provide decision support. Another example would be the integration of a terminology service. It would also be interesting to create a plugin for sharing EHR data using HL-7 or EDI-FACT. A few of these possible components can be seen in figure 8.1.

Last but not least, the practical part of this thesis has been carried out as a proof-of-concept, so it has to be kept in mind that there are several technical topics which have not been addressed, for example a distributed transaction management. Further on, a consistent validation concept as well as a consistent concept for exception handling could be implemented and in addition, a dynamic service lookup registry could be used to integrate new services or change implementations during runtime.

Bibliography

[1] R. Haux, A. Winter, E. Ammenwerth, and B.Brigl. *Strategic Information Management in Hospitals: An Introduction to Hospital Information Systems*. Springer, 2002.

[2] IEEE Architecture Working Group. Ieee std 1471-2000, recommended practice for architectural description of software-intensive systems. Technical report, IEEE, 2000.

[3] Robert T. Monroe, Andrew Kompanek, Ralph E. Melton, and David Garlan. Architectural styles, design patterns, and objects. *IEEE Software*, 14(1):43–52, 1997.

[4] Garlan D. Abowd G., Allen R. Using style to give meaning to software architecture. In *Proceedings of SIGSOFT '93: Foundations of Software Engineering*, volume 118 of *Software Engineering Notes*, pages 9 – 20, 1993.

[5] Florian Leiner. *Medizinische Dokumentation*. Schattauer, 2006.

[6] C.P. Waegemann. Current status of epr development in the us. In *Proceedings of Toward An Electronic Health Record Europe*, pages 116 – 118, 1999.

[7] R.S. Dick, E.B. Steen, and D.E. Detmer. *The computer-based patient record: an essential technology for health care*. National Academy Press, Washington, 1991.

[8] L.L. weed. *Medical records, medical education, and patient care*. Press of Case Western Reserve University, 1969.

[9] Peter Haas. *Medizinische Informationssysteme Und Elektronische Krankenakten*. Springer, 2005.

[10] Judith A. Effken. Different lenses, improved outcomes: a new approach to the analysis and design of healthcare information systems. *I. J. Medical Informatics*, 65(1):59–74, 2002.

[11] Astrid M van Ginneken. The computerized patient record: balancing effort and benefit. *International journal of Medical Informatics*, 65(2):97–120, 2002.

[12] J Wainer, CJ Campos, MD Salinas, and D Sigulem. Security requirements for a lifelong electronic health record system: an opinion. *The open medical informatics journal*, 2:160–165, 2008.

[13] B. Blobel, K Engel, and P Pharow. Semantic interoperability–HL7 Version 3 compared to advanced architecture standards. *Methods of information in medicine*, 45(4):343–53, January 2006.

[14] Robert H Dolin, Liora Alschuler, Sandy Boyer, Calvin Beebe, Fred M Behlen, Paul V Biron, and Amnon Shabo Shvo. HL7 Clinical Document Architecture, Release 2. *Journal of the American Medical Informatics Association : JAMIA*, 13(1):30–9, 2004.

[15] S. Gaion, S. Mininel, F. Vatta, and W. Ukovich. Design of a domain model for clinical engineering within the HL7 Reference Information Model. In *Health Care Management (WHCM), 2010 IEEE Workshop on*, pages 1 –6, 2010.

[16] T. J. Eggebraaten, J. W. Tenner, and J. C. Dubbels. A health-care data model based on the HL7 Reference Information Model. *IBM Systems Journal*, 46(1):5 –18, 2007.

[17] J. Lyman, S. Pelletier, K. Scully, J. Boyd, J. Dalton, S. Tropello, and C. Egyhazy. Applying the HL7 reference information model to a clinical data warehouse. In *Systems, Man and Cybernetics, 2003. IEEE International Conference on*, volume 5, pages 4249 – 4255, 2003.

[18] Barry Smith and Werner Ceusters. Hl7 rim: An incoherent standard. In *MIE*, volume 124 of *Studies in Health Technology and Informatics*, pages 133–138. IOS Press, 2006.

[19] Comité Européen de Normalisation. TC 251 "Health informatics". online at http://www.centc251.org.

[20] Catalina Martínez-Costa, Marcos Menárguez Tortosa, and Jesualdo Tomás Fernández-Breis. Towards ISO 13606 and openEHR Archetype-Based Semantic Interoperability. volume 150 of *Studies in Health Technology and Informatics*, pages 260–264. IOS Press, 2009.

[21] P. Schloeffel, S. Heard, D. Kalra, D. Lloyd, and T. Beale. OpenEHR - Introducing openEHR, 2006.

[22] T. Beale and S. Heard. OpenEHR - Architecture Overview, 2008.

[23] Andrew Goodchild Mark Gibson. Electronic Health Records and openEHR The openEHR Foundation. online at http://portal.extensia.com.au/.

[24] T. Beale. OpenEHR - ISO 18308 Conformance Statement, 2006.

[25] T. Beale, S. Heard, D. Kalra, and D. Lloyd. OpenEHR - EHR Information Model, 2008.

[26] T. Beale and S. Heard. OpenEHR - Archetype Definitions and Principles, 2007.

[27] T. Beale and S. Heard. OpenEHR - Archetype Definition Language 2.0, 2007.

[28] Peter Schloeffel, Thomas Beale, George Hayworth, Sam Heard, and Heather Leslie. The relationship between CEN 13606 , HL7 , and openEHR. 7.

[29] C Horii. Part Four : A Nontechnical Introduction to DICOM1. *RadioGraphics*, (October):1297–1309, 1997.

[30] Oleg S. Pianykh. *Digital Imaging and Communications in Medicine (DICOM): A Practical Introduction and Survival Guide*. Springer, 2008.

[31] Antoine Rosset, Luca Spadola, and Osman Ratib. OsiriX: an open-source software for navigating in multidimensional DICOM images. *Journal of digital imaging : the official journal of the Society for Computer Applications in Radiology*, 17(3):205–16, September 2004.

[32] Matthew a. Wright, Dennis Ballance, Ian D. Robertson, and Brian Poteet. Introduction To DICOM for the Practicing Veterinarian. *Veterinary Radiology & Ultrasound*, 49:S14–S18, January 2008.

[33] C Carr. IHE: a model for driving adoption of standards. *Computerized Medical Imaging and Graphics*, 27(2-3):137–146, June 2003.

[34] JM Corrigan, LT Kohn, MS Donaldson, SK Maguire, and KC Pike. *Crossing the Quality Chasm: A New Health System for the 21st Century*. National Academy Press, Washington, DC, 2001.

[35] James R. Langabeer. *Health Care Operations Management: A Quantitative Approach to Business and Logistics*. August 2007.

[36] Jinyoul Lee, Keng Siau, and Soongoo Hong. Enterprise integration with ERP and EAI. *Communications of the ACM*, 46(2):54–60, February 2003.

[37] Chonyacha Suebsin and Nathasit Gerdsri. Technology Adoption : A Case Study of ERP Implementation in One of Healthcare Organizations in Thailand. 2010.

[38] Khaled Alfawaz. Critical Success Factors in Enterprise Resource Planning Implementation: A Case Study in Saudi Arabia Hospital.

[39] D.A. Brachos Kostopoulos, K.C. and G.P. Prastacos. Determining factors of ERP adoption: An indicative study in the Greek market. *International Engineering Management Converence*, pages 287–291, 2004.

[40] Nuri Basoglu, Tugrul Daim, and Onur Kerimoglu. Organizational adoption of enterprise resource planning systems: A conceptual framework. *The Journal of High Technology Management Research*, 18(1):73 – 97, 2007.

[41] K.J. Trimmer, L.D.K. Pumphrey, and C. Wiggins. ERP implementation in rural health care. *Journal of Management in Medicine*, 16:113–132, 2002.

[42] Keith J. Dreyer. *PACS: A guide to the digital revolution*. Springer, 2006.

[43] H. K. Huang. *PACS and Imaging Informatics: Basic Principles and Applications*. John Wiley & Sons, 2010.

[44] Roger Bauman, Guenther Gell, and Samuel Dwyer. Large picture archiving and communication systems of the world—part 1. *Journal of Digital Imaging*, 9:99–103, 1996.

[45] Roger Bauman, Guenther Gell, and Samuel Dwyer. Large picture archiving and communication systems of the world—part 2. *Journal of Digital Imaging*, 9:172–177, 1996.

[46] Herman Oosterwijk. *PACS Fundamentals*. O Tech, 2004.

[47] C. Laske. Legal issues in medical informatics: A bird's eye view. I O S PRESS, 1996.

[48] Bernd Bobel. *Analysis, Design and Implementation of Secure and Interoperable Distributed Health Information Systems*. IOS Press, 2002.

[49] RH Smuckler. Health care information: Access and protection. 1994.

[50] Ross J. Anderson. A security policy model for clinical information systems. In *IEEE Symposium on Security and Privacy*, pages 30–43. IEEE Computer Society, 1996.

[51] Bryden Darley, Antony Griew, Kathryn McLoughlin, and John Williams. *How to Keep a Clinical Confidence*. HMSO, 1995.

[52] Ian Sutherland. Relating bell-lapadula-style security models to information models. In *CSFW*, pages 112–123, 1988.

[53] Qingui Xu and Guixiong Liu. Configuring clark-wilson integrity model to enforce flexible protection. In *CIS (2)*, pages 15–20. IEEE Computer Society, 2009.

[54] Ren Hong-min, Zhang Jing-zhou, and Yan Zhi-ying. Research on regional his interconnection architectures based on electronic medical record. In *Software Engineering, 2009. WCSE '09. WRI World Congress on*, volume 1, pages 84 –88, May 2009.

[55] J.A. Maldonado, M. Robles, and C. Cano. Integration of distributed healthcare information systems: application of CEN/TC251 ENV13606. In *Engineering in Medicine and*

Biology Society, 2001. Proceedings of the 23rd Annual International Conference of the IEEE, volume 4, pages 3731 – 3734, 2001.

[56] Haiyan Chen, Shigang Qin, Jianxun Liu, and Jian Cao. SOA-Enabled Health Information Integration Platform (HIIP): A Case Study. In *Semantics, Knowledge and Grid, 2009*, pages 384 –387, 2009.

[57] J.K. Zhang and W. Xu. Web Service-based Healthcare Information System (WSHIS): A Case Study for System Interoperability Concern in Healthcare Field. In *Biomedical and Pharmaceutical Engineering, 2006. ICBPE 2006. International Conference on*, pages 588 –594, 2006.

[58] E. Vasilescu and S.K. Mun. Service Oriented Architecture (SOA) Implications for Large Scale Distributed Health Care Enterprises. In *Distributed Diagnosis and Home Healthcare, 2006. D2H2. 1st Transdisciplinary Conference on*, pages 91 –94, 2006.

[59] Weiping Wang, Mingming Wang, and Shijun Zhu. Healthcare information system integration: a service oriented approach. In *Services Systems and Services Management, 2005. Proceedings of ICSSSM '05. 2005 International Conference on*, volume 2, pages 1475 – 1480, 2005.

[60] Meg Murray and Meg C Murray. An Initial Investigation of Web Services in Healthcare. *Information Systems*, 2003.

[61] Santiago Comella-Dorda, John C. Dean, Edwin Morris, and Patricia Oberndorf. A process for cots software product evaluation. In *In proceedings of the International Conference on COTS-Based Software Systems ICCBSS*, pages 86–96, 2002.

[62] K. Ven, I. Verelst, and H. Mannaert. Should you adopt open source software? *Software, IEEE*, 25(3):54 –59, 2008.

[63] B. Kitchenham, S. Linkman, and D. Law. DESMET: a methodology for evaluating software engineering methods and tools. *Computing & Control Engineering Journal*, 8(3):120, 1997.

[64] S.L. Pfleeger. Experimental design and analysis in software engineering. *Annals of Software Engineering*, 1(1):219–253, 1995.

[65] Xudong Lu, Huilong Duan, Haomin Li, Chenhui Zhao, and Jiye An. The architecture of enterprise hospital information system. *IEEE Engineering in Medicine and Biology Society*, 7:6957–60, January 2005.

[66] Andrew Forward and Timothy C. Lethbridge. The relevance of software documentation, tools and technologies: a survey. In *Proceedings of the 2002 ACM symposium on Document engineering*, DocEng '02, pages 26–33, 2002.

[67] Andrew M. St. Laurent. *Understanding Open Source and Free Software Licensing*. O'Reilly Media, Inc., 2004.

[68] Pauline Ratnasingham and Paul Pavlou. The Role of Web Services in Business to Business Electronic Comerce. *Information Systems*, 2002.

[69] Vandana Singh. *Knowledge creation, sharing and reuse in online technical support for open source software*. PhD thesis, Champaign, IL, USA, 2008.

[70] Nicole Serrano and Jose Mare Sarriegi. Open Source Software ERPs: A New Alternative for an Old Need. *IEEE Software*, 23:94–97, 2006.

[71] A. Vazquez, S. Bohn, M. Gessat, and O. Burgert. Evaluation of Open Source DICOM Frameworks, 2006.