



Leopold-Franzens-University Innsbruck



Institute of Computer Science
Quality Engineering

Integrated Security Analysis of the Open Source
Health Information System Elexis
Master Thesis

Author
DI (FH) Descher Marco

supervised by
Prof. Ruth Breu
Dr. Frank Innerhofer-Oberperfler

Mäder, July 13, 2010

Abstract

Data forms the basis for information, leading to knowledge being applied to reach decisions. Especially in the medical field, the security and quality of data leading to decisions is of utmost importance.

The thesis at hand is both the extension of an existing thesis conducted at the University of Innsbruck concerning the security of the Elektronische Gesundheitsakte (ELGA) and a data-driven approach to an integrated security analysis of the electronic medical record within a practitioners practice.

To this end the model-based analysis process ProSecO is extended to consider analysis on a specific component (that is applications) with respect to a certain information asset. To underpin its relevance, the selection of the specific data or information asset to be investigated is carried out by means of business impact analysis, selecting the asset which on loss or degradation would implicate the highest financial and non-financial loss.

The thesis also extends the analysis process to take into account threats to semantic integrity of information, which may occur on uneducated transfer of data from analog to digital resources, and points out specific aspects found within the local domain not yet specifically considered within ProSecO.

The modified analysis process is applied in a case study settled within the health-domain to showcase its applicability and the resulting output. For this purpose the analysis is focused on the open source health information system *Elaxis* and its management of a patients electronic medical record.

Affidavit

I hereby declare that I have written this thesis by my own. Furthermore, I confirm that no other sources have been used than those specified in the thesis itself. This thesis, in same or similar form, has not been made available to any audit authority yet, nor has it been published.

Ich erkläre hiermit an Eides Statt, dass ich die vorliegende Masterarbeit selbständig angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher weder in gleicher, noch in ähnlicher Form einer anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

DI (FH) Descher Marco, Mäder, Juli 2010

Contents

1	Introduction	1
1.1	Challenge	1
1.2	Domain of Investigation	2
1.3	Objects of Investigation	2
1.4	Methodology	2
1.5	Structure	3
2	Background and research question	5
2.1	Asset Data	5
2.2	Data Classification	6
2.3	Data Governance	8
2.4	Model-based security analysis	10
3	Fitting ProSecO to the aim of investigation	15
3.1	Bridging application and software layer	15
3.1.1	Approaching white-box security testing	17
3.1.2	(Simplified) White-box security testing	18
3.1.3	Integration in ProSecO: The implementation view	19
3.2	Introducing Quality objectives	22
3.3	Local domain specifics	24
4	Case study: Medical Practice	27
4.1	Data objects vivid in a medical practice	28
4.2	Data Classification	30
4.3	Modeling the sample workflow	31
4.3.1	Business layer	31
4.3.2	Application layer	33
4.3.3	Software layer	34
4.3.4	Physical layer	38
4.4	Deriving security and quality objectives	39
4.5	The sample workflow inter-layer dependencies	41
4.6	Security and quality requirements engineering	41
4.7	Threat and risk analysis	44

4.7.1	Understand the rationale	45
4.7.2	Compare rationale with design principles	45
4.7.3	Model the relevant aspects	45
4.7.4	Compare model with patterns and rules	45
4.8	Mitigation	54
4.9	Report	56
5	Summary	59
5.1	Achievement of objectives	59
5.2	Reflection on the analysis method	60
5.3	Closing	61
	List of abbreviations	63
	Bibliography	69

Chapter 1

Introduction

Data forms the basis for information, leading to knowledge being applied to reach decisions. Especially in the medical field, the security and quality of data leading to decisions is of utmost importance.

Be it either a hospital or a physician in private practice, paper records and/or Electronic Medical Record (EMR) systems are applied to manage the data that form the base for diagnoses and treatment of diseases.

The security of patient-related data is imperative, and tight regulations are set on this according to different laws and standards (e.g. Datenschutzgesetz (DSG) 2000 [Jah01] or ISO 27799 [ISO08]). These laws are already considered in the infrastructure and process analysis as to conform to legislative compliance requirements.

The quality of the data the diagnosis is based on is, however, of more immediate importance to the patient and yet this is not considered within the ProSecO analysis model which will be subsequently applied.

1.1 Challenge

The aim of this thesis is to conduct an integrated, data-driven analysis of security and quality objectives starting from an enterprise architecture view top-down to the software architecture view.

While antecedent work in [Mit08] focused on the global context of handling a patients Electronic Health Record (EHR), this elaborates on the specific handling of the EMR within a practitioners practice.

1.2 Domain of Investigation

The investigation for the use case has been carried out within a cardiologists practice. Certain assumptions and facts are taken from this domain, the approach itself is however applicable to any kind of medical practice.

1.3 Objects of Investigation

As the aim is to provide an integrated security analysis, the object of investigation is the medical practice and its business processes.

Specific focus is laid on the integrated analysis of software components. Exemplarily, in the treated case study, the open source Health Information System (HIS) *Elexis* [WE10], which is an abbreviation for *Elektronische Praxis* founds the base for the security analysis. Elexis is selected due to the fact that it is the only free and open-source EMR software known to the author which is in practical use and accessible in a real business operation environment.

Elexis is based on the Eclipse Rich Client Platform (RCP). The analysis of Elexis is confined to its base packages and to version 2.0 available in the sourceforge repository <http://sourceforge.net/projects/elexis/>. The respective used base packages (w.r.t. to the repository location) are:

- `elexis/branches/2.0/elexis`
- `elexis/branches/2.0/elexis-utilities`
- `elexis/branches/2.0/postgresql-adapter`

1.4 Methodology

On one hand, the thesis is a continuation of [Mit08, chapter 4.2.2.3] further inspecting the local technical model of a physician in a private practice. To this end, the global view is excluded, as the EHR does merely exist in global context, and the focus is laid on the EMR.

On the other hand, a data driven approach is taken. The business value of data occurring within the practice, employing a business impact

analysis, is evaluated. This reveals the fact that the EMR is the most important data asset within a physicians practice.

1.5 Structure

This thesis is separated into five big chapters

- *Chapter 1: Introduction*
- *Chapter 2: Background and research question* lays the theoretical foundation for the tools and methods employed within the subsequent chapters.
- *Chapter 3: Fitting “ProSecO” to the domain of investigation* presents the required enhancements and modifications to ProSecO for realizing the security analysis on the local layer.
- *Chapter 4: Case Study* employs ProSecO and its enhancements and modifications on a physicians practice.
- *Chapter 5: Summary and Results* provides a roundup on the work done.

Chapter 2

Background and research question

This chapter lays the theoretical background for the chapters to come. As for the paper to be relatively self-consistent, a knowledge foundation for the topics to come needs to be provided.

- Asset data
- Data classification and valuation
- Data Governance
- The ProSecO security analysis method
- Security and design patterns and principles

2.1 Asset Data

While it is relatively easy and common practice to value tangible assets within an enterprise (e.g. during the process of accounting depreciation), intangible objects are mostly eluding a valuation and hence are not directly seen by the Chief Executing Officer (CEO) or more generally, the persons in charge of and responsible for business decisions.

It is, however, very important to see data existing within the enterprise as a company key asset, as it forms the basis for decisions and actions of every member of the resp. company.

Several disciplines, such as Master Data Management (MDM) or Data Lifecycle Management (DLM), besides their technical and management focus, aim towards creating awareness on the value of data and its life-cycle to non-technical professionals.

A primer on data types

Different opinions resp. different accesses exist on the types of data existing. The aim hence is not to create or introduce a full-fledged taxonomy, but merely to find a crude way of arranging the data elements found.

The result of a short literature survey on data type arrangements can be seen in table 2.1. So, apart from a more specific arrangement, there seems to be a general allocation to three different types of data.

Microsoft [WH06]	Wikipedia Stammdaten	Oracle [But09]
Master Data	Master Data	Master Data
Transactional	Transaction Data	Transactional data
Metadata	Inventory Data	Analytical Data
Hierarchical		
Unstructured Data		

Table 2.1: Arrangement of data types found in the resp. literature.

The arrangement of data into the resp. types can be based upon life-time, existential dependence, extent of modifications, content, process-oriented or temporal aspects.

2.2 Data Classification

To classify the identified data types, a classification scheme first has to be selected and subsequently acted upon. There are several schemes, descending from different focus areas.

Data can be e.g. classified according to security levels [Bel05] (e.g. unclassified, sensitive-but-unclassified, confidential, secret and top secret), integrity levels [San94], quality levels [MBEH09] and so on. The classification scheme again is dependent upon the domain of investigation, and the semantics to be achieved.

In the domain of investigation relevant within this paper, a classification according to business value of data is required, as to identify the *most valuable data assets*.

It is yet an open discussion (cf. eg. [VM04]) whether it is possible to quantify the business value of (master) data, such as the possible quantification of tangible assets. It is, however, possible to classify the impact on the business, due to loss or degradation of specific data assets, applying a simplified form of Business Impact Analysis (BIA).

So to bridge the gap, and define the term value in our specific domain, *value* is determined to be the level of impact-on-loss or degradation, delivered through the BIA.

Valuation through Business Impact Analysis

In [SFBH⁺06, chapter 6.2] the Asset Valuation pattern is presented, described by the following statement: *Asset valuation helps you to determine the overall importance an enterprise places on the assets it owns and controls. Loss or compromise of such assets may result in anything from hard costs, such as fines and fees, to soft costs due to loss of market share and customer confidence.*

A basic assumption behind BIA (which is another word for asset valuation) is hence that *every component of the organization is reliant upon the continued functioning of every other component, but that some are more crucial than others and require a greater allocation of funds in the wake of a disaster* [Mil10].

These definitions are adapted to value the identified data types, bearing the key question: The loss or degradation of which data type does inflict the most serious business impact?

As stated, there are two general kinds of business impact:

- **Financial:** Loss of revenue, Loss of shareholder value, Penalties, Bad debts, Additional operating costs
- **Non-Financial:** Reputational loss, Loss of operational capacity, Customer-service, Regulatory/legal, Loss of market share, Loss of quality, Brand tarnish, Environmental, Contractual, Staff Moral, Political

Different sources recommend different ways for realizing qualitative valuations of the assets, [MBEH09, Chapter3.3.9] recommends asking and evaluating the following key questions:

1. What percentage change to revenue would occur?
2. What percentage change to costs would occur?
3. What risk exposures might occur, and what would be the potential financial impact?

[SFBH⁺06, chapter 6.2] however presents a more structured approach, where four steps systematically determine the overall value of the identified assets. The aforementioned questions may be used to determine the outcome of question three:

1. Determine the security value.
2. Determine the financial value.
3. Determine the impact on business.
4. Determine the overall value and build an asset valuation table.

Evaluation and treatment of the presented questions and systematical steps lead to an “Information asset valuation table”. Referring to the complexity of the area of investigation, asset valuation can be either approached in a pragmatic or a more elaborated way.

An elaborated asset valuation would include a quantitative assessment of assets, threats and risks. A quantitative assessment using A Process Model for Security Engineering (ProSecO) is presented in [BIOY08]. A more pragmatic approach, however, does only apply a qualitative assessment, by indicating a certain classification (e.g. low, medium, high).

2.3 Data Governance

The topic of Data Governance (DG) is introduced in order to provide the basic notion for Data Quality (DQ) as it will become an integral part of the overall analysis procedure. The information presented within this section is an excerpt of [Des10].

DG is a structure for making decisions about the management of data and all aspects contributing to the business value of data. So while IT Governance is concerned with the management of systems, processes and so on, DG cares about the management of an enterprises data processed by these.

DG defines several concepts, metrics and maturity models. One of the most important concepts, with respect to this thesis, is the notion of data quality and data stewardship.

A glimpse of data quality and data stewardship

Data quality can be set according to different criteria, an overarching definition however states that the quality of data is determined by its realized objective of:

- Accuracy
- Reliability
- Completeness
- Appropriateness
- Timeliness
- Credibility

It can also be defined as the grade of *truth* that lies within the information, respective to the empirical meaning.

Data Stewardship is the ownership of data by a certain person called the *Data Steward*. It is the Data Stewards responsibility to increase or maintain the data quality of its consigned data sets. Generally there exist two types of Data Stewards:

- *Functional Data Stewards* are members of a specific department. They are experts in a specific business domain (e.g. Production, Marketing, ...). The Functional Data Steward orchestrates the data quality measures for his area of responsibility. He knows about the meaning of data and hence is the strongest contributor to data quality.

- *Technical Data Stewards* the counterpart to the Functional Data Stewards, are mostly IT-related persons. They cope with questions about the data architecture and the necessary infrastructure to realize data quality. They know about the appropriate IT data formats and structures.

2.4 Model-based security analysis

Model-based security analysis [BHLOW08] paves the way for a traceable process from requirements engineering down to the high-level realization of secure solutions.

Several model-based security analysis methods like CORAS [dBHL⁺07] and Octave [AD02] exist. The topic of this thesis is, however, not an evaluation resp. an inspection of analysis methods, but the extension and application of the *ProSecO* method.

ProSecO itself is an extension of the Model Driven Architecture (MDA) approach to integrate security requirements at the abstract level into the Platform Independent Model [HB08, p50].

ProSecO: A Process Model for Security Engineering

ProSecO originally encompasses security analysis on business processes, dispersed over different institutions. So both a global and a local system meta-model are provided.

The meta-model elements, cf. figure 2.1 and figure 2.2, are classified along two orthogonal categories:

- **Level of Interaction:** The *Global View* describes aspects related with the interaction of different stakeholders, the *Local View* describes aspects related to the behavior and structure of a specific stakeholder.
- **Level of Abstraction:** Three levels of abstraction are distinguished in the meta model, the *business level* describes business requirements, whereas the *application level* and the *technical level* are concerned with the executable solution.

Security analysis is executed on specific enterprise models or selected business processes out of this enterprise model. The functional elements

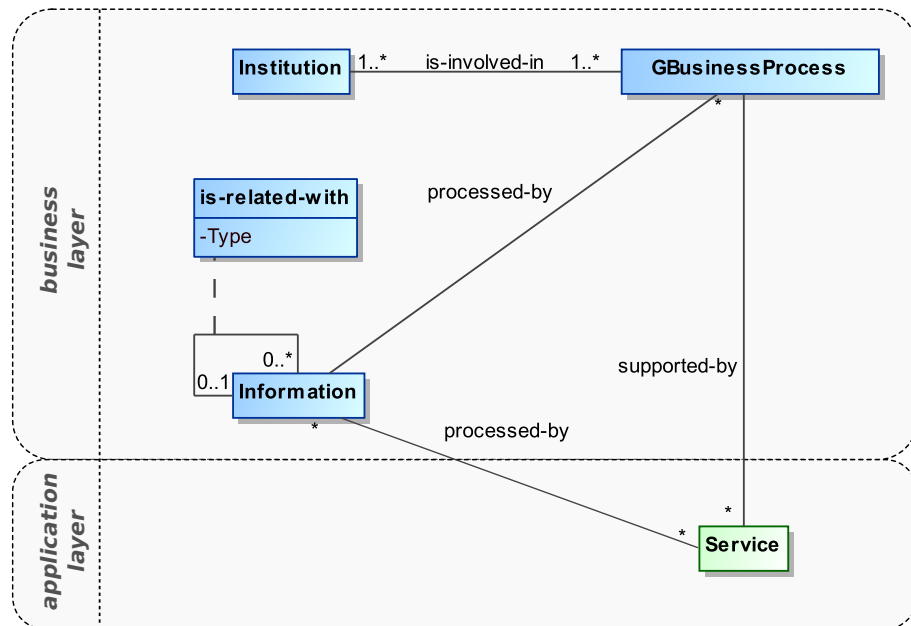


Figure 2.1: ProSecO global system meta-model (or global view). Figure from [BHIOW08]

are evaluated applying the ProSecO security meta-model, which is shown in figure 2.3.

This (figure 2.3) security meta-model has to be read as follows: A specific element of the global or local meta-model (`ModelElement`) has to comply to a certain business security objective (`BusinessSecurityObjective`) (e.g. “Data must not leave the enterprise”). A business security objective generates layer specific security requirements (`SecurityRequirement`) to be enforced. Each security requirement is affected by a certain risk (`Risk`) originating from a threat (`Threat`). This risk is addressed by a specific security control (`SecurityControl`) which presents a measure or safeguard that has been put in place to mitigate the identified risk. States, announcing the current state of the security analysis, are allocated to both business security objectives (`BSOState`), security requirements (`SecurityRequirementState`) and risks (`RiskState`).

At each point of time during the security analysis, the system is described by a set of interrelated model elements, where these model elements either adhere to the functional meta-model types of figures 2.1,

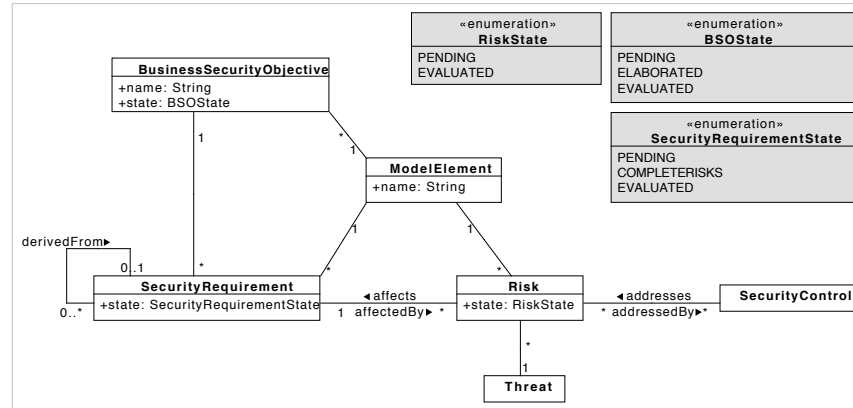


Figure 2.3: The ProSecO Security Meta Model. Figure from [BHIOW08]

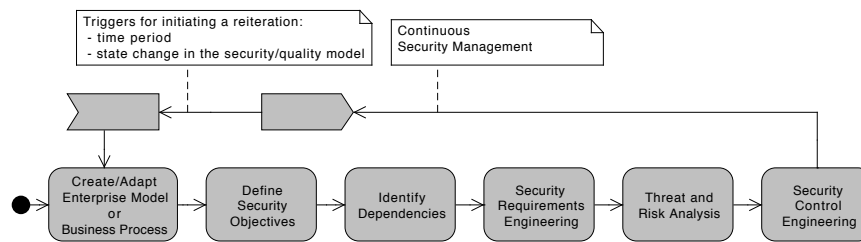


Figure 2.4: The ProSecO Analysis Process. Figure from [BHIOW08]

whole security model, but on sub-models, where a sub-model is called a security domain.

A detailed explanation of the ProSecO security analysis method, can be found in [BHIOW08].

Chapter 3

Fitting ProSecO to the aim of investigation

In the treated domain of investigation we do not consider the global system meta-model, as the relevant analysis is only executed on entities local to a single stakeholder. So the security model applied will only adhere to either the local system meta model types of fig. 2.2 and to the security meta model of fig. 2.3.

It is required to alter some parts of ProSecO to accommodate for the (treated) security analysis domain and the deeper analysis of the software managing the crucial data (Elexis).

To realize this the following adaptations will be dealt with:

- Bridge the gap between enterprise architecture and software architecture to extend the security model into the EMR system.
- Augment the ProSecO analysis process to account for quality objectives.
- The local technical model needs to consider more fine grained technical solutions and their specific characteristics.

3.1 Bridging application and software layer

ProSecO covers the enterprise architecture layers security objectives by annexing requirements, threats and measurements to the single model

elements. Part of this enterprise architecture diagrams are the components.

A component is an identifiable part of a larger program or construction. Usually, a component provides a particular function or group of related functions [Wha10].

Components (applications) hence can be modeled and analyzed in a deeper fashion, by focusing on certain functional parts of the implementation.

Figure 3.1 shows an adapted meta-model including a deeper analysis of single components, respective applications. The meta-model was developed within the Secure Change project [BMIO⁺10]. Secure change does however focus on integrating change management into the overall software developing process. The (Secure Change) project treats methods of software security testing, these are however oriented on formal methods.

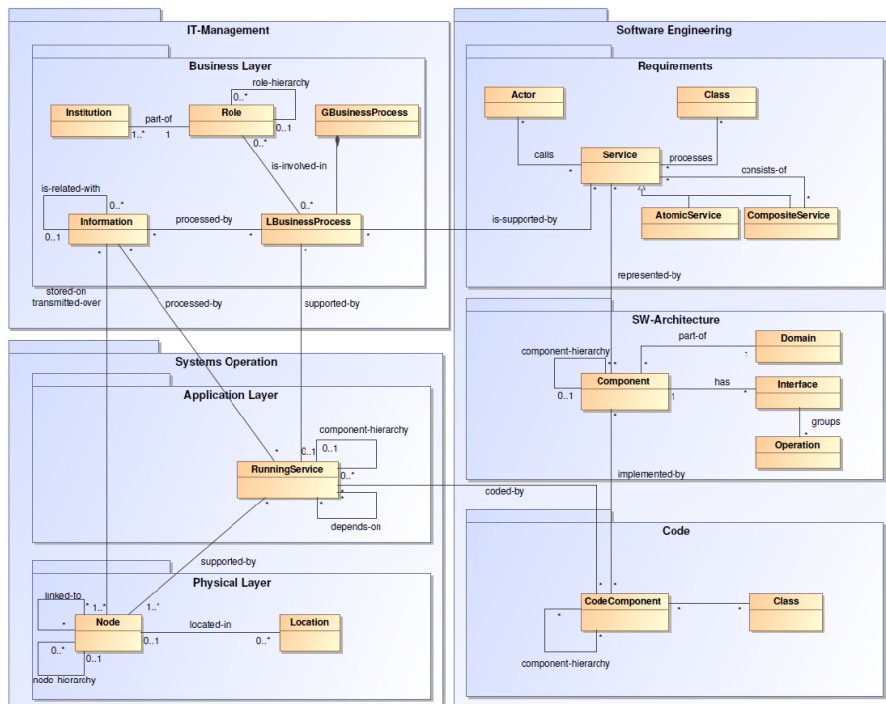


Figure 3.1: Functional System Meta Model of the Secure Change project [BMIO⁺10].

ProSecO does not generate quantified assertions about the vulnerability or security of a certain element within its model, it however presents a qualified estimation of security against a certain threat or risk. There are several projects in computer science working on provable correct programs. To fit into the ProSecO context, however, such an approach is far beyond the timely capability and saleability of a generic security assessment. Accounting for this fact, a straightforward and yet effective way to security assess the security relevant implementation of a certain aspect or data item within a component has to be found.

[MP04] describes several aspects of software security testing, basically describing two approaches:

- **Black-box testing**; analyzing a running program by probing it with various inputs – source code not available
- **White-box testing**; analyzing and understanding source code and design – source code available

These approaches are not mutually exclusive. Methods of black-box testing may even be incorporated into the white-box testing procedures by e.g. extending JUnit testing with fuzzy testing methods.

The authors of [MP04] go on to describe that vulnerabilities typically fall into two categories – bugs at the implementation level and flaws at the design level. Design level flaws include error handling in object-oriented systems, object sharing and trust issues, unprotected data channels, etc. Implementation bugs relate to errors like race conditions or unchecked boundaries.

3.1.1 Approaching white-box security testing

As the aim of investigation is an open source program, hence making the source code available, white-box testing is the way to go. As already denoted white-box testing involves analyzing and understanding the source code and the design.

Good design and clean implementation can be reached by relying on the following integral software development concepts:

- **Design patterns** [MW06] and **security patterns** [SFBH⁺06] are *solutions to problems arising within a specific context. They*

describe a particular recurring design problem that arises in specific design contexts, and present a well-proven generic solution for it. The respective solution consists of a set of interacting roles that can be arranged to form multiple concrete design structures, as well as a process for creating any particular structure. [RFMPG06]

- **Design principles** describe the design and implementation of security mechanisms. In [Bis03, chapter 13] eight patterns are presented, that draw on the overarching principle of simplicity and restriction:
 - Principle of least privilege
 - Principle of fail-safe defaults
 - Principle of economy of mechanism
 - Principle of complete mediation
 - Principle of open design
 - Principle of separation of privilege
 - Principle of least common mechanism
 - Principle of psychological acceptability
- **Implementation rules** and **management rules** target at the conversion from design to the implementation. [Bis03, chapter 29] presents 18 implementation and 8 management rules. Sticking to these rules during the process of implementation is likely to reduce bugs to a minimum.

3.1.2 (Simplified) White-box security testing

To come up to the requirements of the testing scenario, the following parts have to be analyzed: the security mechanisms itself and the application of the available security mechanisms by the aspect to be analyzed. The approach for such an analysis, with respect to the analysis context, is as follows:

1. **Understand the rationale** – An understanding of the rationale of both the security system and the aspect to be analyzed is prerequisite. It is necessary to understand the intention of the programmer.

2. **Compare rationale with design principles** – The rationale has to be compared against a set of design principles and possibly best practices, as to expose flaws within the basic rationale (e.g. the principle of fail-safe defaults states that, *unless a subject is given explicit access to an object, it should be denied access to that object*. Does the rationale support this design principle?)
3. **Model the relevant aspects** – A model of the security system, environment of aspect to be analyzed and their interconnection has to be created, considering the following viewpoints:
 - *Component internals* – Internal working of the component (e.g. unintentionally derivable classes)
 - *Component persistence* – Method of persisting information (e.g. mapping objects to a relational database, upstream caches)
 - *Component configuration* – Configuration method of the component (e.g. configuration files and possible misconfigurations, correctness of configuration)
 - *Component interaction* – Interaction with other components (e.g. Java Database Connectivity (JDBC) link properties)
4. **Compare model with patterns and rules** – The code modeled has to be compared against existing design patterns and rules of implementation. Although there exist more formal methods, like [BM03], an informal linking approach, relying on the experience of the respective analyst, does fit the ProSecO approach more closely.

3.1.3 Integration in ProSecO: The implementation view

Combining an element of the information view with an element of the application view results in a implementation view (cf. figure 3.2).

The implementation view is split into three parts:

- **Representation of information in component:** In the object-oriented programming paradigm (which the analysis is confined to), data (at runtime) is represented as instantiation of classes, populated with values for variables and constants. So to analyze

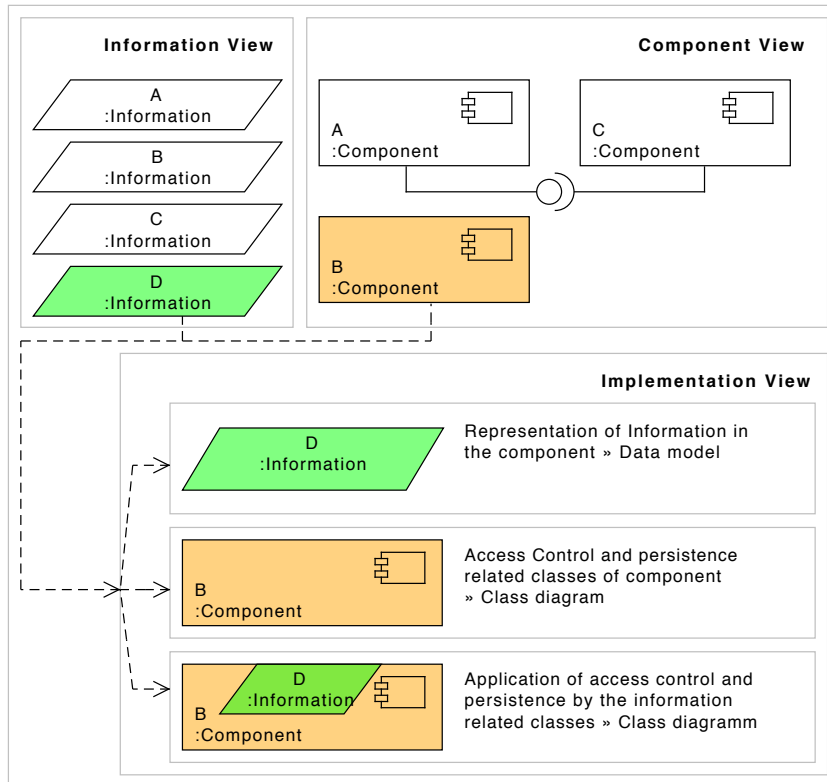


Figure 3.2: The combination of a specific information element with a component element results in an implementation view.

a specific data asset in full context, we need to generate an understanding on the implementation of the data within the application. The task hence involves creating a data model of **D:Information** with corresponding multiplicity and relation in **B:Component**.

- **Access control and persistence related classes of component:** As the security within **B:Component** is based on the security measures provided by the component itself, an investigation of the respective implementation has to be included. Hence the classes of **B:Component** representing the access control and persistence need to be modeled to understand the components self protection.
- **Application of access control by the information in the component:** Finally the application of the access control by the

specific information implementing classes needs to be determined. This determines whether the information adequately protects itself given the access control of `B:Component`. Ideally each method that performs a Create, Read, Update or Delete (CRUD) operation on its represented information request for permission on this operation.

Consequently the implementation view does inherit its security objectives and requirements from the information and component element it is derived from.

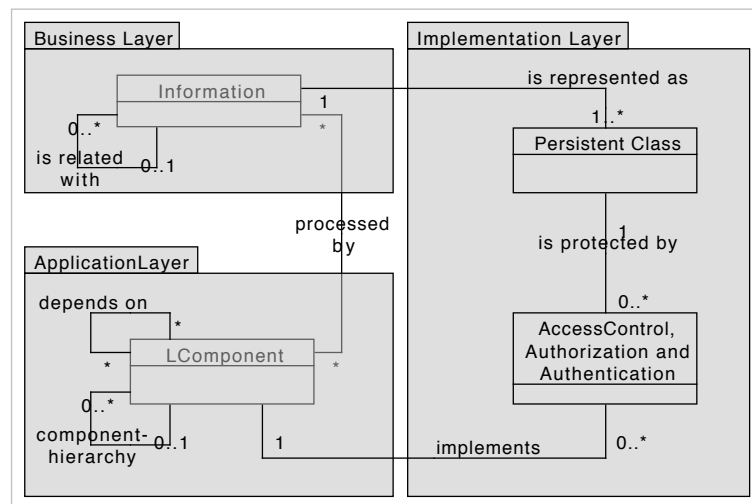


Figure 3.3: Adaption to the ProSecO meta model to consider the implementation layer.

Embedding the implementation layer into the local ProSecO meta model results in the modification presented in figure 3.3. Information is represented as a set of persistent classes, where the relation of these classes with the information can be separately analyzed employing a data model. The component (possibly) implements access control, authorization and authentication. This in turn protects the persistent classes against unauthorized access.

3.2 Introducing Quality objectives

The probability of data quality degradation rises, in every data treating process, when the meaning of the data is not entirely clear to the processor, and/or the data source provides inferior information (e.g. a blurred piece of paper, or a noisy voice recording).

So if, for example, a non-domain-expert clerk is assigned to enter blurred laboratory values from a legacy chemistry lab devices printer into the EMR system, mistakes are prone to occur. These mistakes however are not as probable to a domain-expert clerk, as she knows about the semantics of the data, and hence is able to correct the information on processing (cf. chapter 2.3, Functional Data Steward).

Such a degradation of data quality poses a risk to the enterprise, especially when seen with relation to medical values. By embedding this fact into the security process, we meet concerns relating integrity and *semantic integrity* of the data.

Semantic integrity threats occur on data exchange including a context interchange and a required mediation or on transfer of information leading to a “Medienbruch”. [MMM⁺01] describes the terms in relation to information exchanged across organizational boundaries. An example found for the aim of investigation is the exchange of laboratory values: Each laboratory device has a certain normal range for a specific test (depending on male or female). If a resulting laboratory value is hence sent to a receiver without the device specific normal range, semantic integrity is lost, while the value itself is still of integrity.

Figure 3.4 extends the original local meta model, as presented in figure 2.2 to provide for the aforementioned data quality concerns. This is done by introducing the concept of an *analog physical node*, in contrast to the original node considered to be of *digital kind* (an analog node of type paper has already been applied in [IOB06a], not applying however an analog decoration).

To consider the loss of data quality on transfer from an analog to a digital representation form (i.e. physical representation), the notion of a directed flow of information has to be introduced. Again this is accommodated by applying stereotypes.

Summing up, figure 3.4 extends the original local meta model in so far,

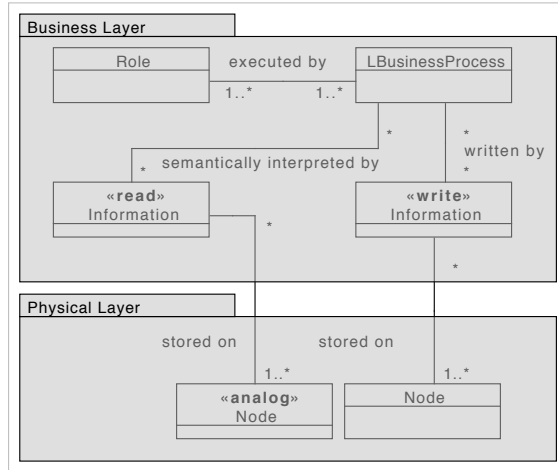


Figure 3.4: Adaption to the ProSecO meta model. The concept of an analog node, and directed flow of information is introduced.

as to provide for the data quality concerns arising by a business process translating information contained by an analog physical node into a digital representation. Considering this representation, a quality threat is allocated to a business process, semantically interpreting information from an analog node to be transferred into an information stored on a digital physical entity.

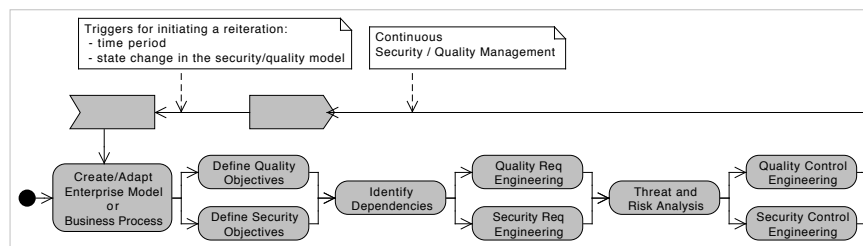


Figure 3.5: The ProSecO Analysis Process, adapted to include quality analysis. (cf. [HB08, p82])

Figure 3.5 presents the modifications to the ProSecO analysis process to include quality analysis. Concurrent to the definition of the respective security objectives, for example, a set of quality objectives can be issued.

The modifications to ProSecO presented, suffice for the quality concerns raised within the aim of investigation. In no way the aim of the thesis is to provide a full integration of data quality concerns into the ProSecO security process.

3.3 Local domain specifics

Focusing on the local domain, security requirements need to be concretized, as on the single processes happening on local physical resources, measures are to be realized.

One has, however, to set a judicious limit on the applied level of granularity. A security measurement “Encrypting the filesystem” required by a security threat could easily be further elaborated by deepening into *security realization requirements* like e.g. “Selection of a sufficiently secure (in terms of cryptological requirements) encryption method”.

Confining to measurements feasible within a business environment (w.r.t. time, monetary and human resources) one has to stick, however, to a reasonable set of local domain specific security requirements.

A full coverage on integration of these requirements on the infrastructure level and its processes, is beyond the scope of this thesis. It is, however, a concern of the author to mention several open security related questions, to be possibly treated within subsequent work.

- *Virtual hosts* reside between the physical and the component layer, depending on their constitution. This possess an entire new threat respectively risk scenario, as inter-virtual-machine attacks need to be considered.
- *Aspects of real computer system hardware* are not yet considered. A part of security is Availability. So the redundancy of certain systems and possibly failover methods need to be adopted as concrete security measurements.
- *Aspects of information security on local networks* are partly considered by the mentioning of **SR_31** (Adaption of the IT security measures) in [Mit08], need however to be more elaborated on.

- *Backup systems* form an integral part of security. Respective business processes for backup and restore need to be created and evaluated, to reduce downtime to a minimum.
- *Configuration management* makes a big part of secure software. Several applications resp. components are secure by themselves, may expose security vulnerabilities due to misconfiguration however. Especially in the are of web services, configuration management needs to be considered within the security analysis.

Chapter 4

Case study: Medical Practice

In [Mit08] the local functional model of a general practitioner is touched. The thesis, being however focused on the Austrian EHR *ELGA* (Elektronische Gesundheitsakte), stops at this point, as it centers the ELGA respective global model.

This case study extends at this point by bridging the gap from the global EHR to the local EMR (where a consecutive number of EMR form a persons EHR) but especially takes into account the environment of a practitioner and its data assets. Hence the decision to analyze especially the management of the EMR is not directly made, but derived from the fact that the recordings of a patient are most relevant for the practitioner to make (diagnostic) decisions. Nevertheless certain findings from [Mit08] form the input for security related analysis concepts (e.g. the Business Security Objective (BSO)).

The case study is using a data-driven approach which is carried out according to the following steps, where the theoretical foundation has been laid in chapter 2 and the method has been adapted in chapter 3. As the method is data asset driven, we have two additional steps, before entering the analysis process circle (cf. figure 3.5 on page 23). Additionally a summary resp. report on the analysis method will be given.

1. Identify the data objects (chapter 4.1)
2. Classify the identified data objects (chapter 4.2)

3. Create/adapt enterprise model or business process (chapter 4.3)
4. Define security and quality objectives (chapter 4.4)
5. Identify Dependencies (inter layer) (chapter 4.5)
6. Quality and security requirements engineering (chapter 4.6)
7. Threat and risk analysis (chapter 4.7)
8. Quality and security control engineering (chapter 4.8)
9. Report on the analysis method (chapter 4.9)

4.1 Data objects vivid in a medical practice

As mentioned during the introduction, the specific domain of investigation in the case study is a cardiologist and internal specialist, so the data types identified are in part strongly related to the specific medical domain. Nevertheless the method of identification can be applied to any general or specific practitioner.

Table 4.1 presents the data objects identified during an extended interview with personnel in the private practice and the scanning of the local computer systems and infrastructure. The identified data is compartmented into four empirically derived types, namely:

- **Raw Measurement Data:** Data, generated through tests delivering quantified values
- **Interpreted Data:** Data generated through the practitioner's decision and knowledge
- **Management Data:** Data concerning the surroundings and the procedures of the private practice
- **Unstructured Data:** Miscellaneous data that is referred to

These data, as any data, are interconnected through business process being executed within the enterprise. To allow proper assessment resp. classification of the data types, additional information about the interconnection of these data needs to be gained.

Raw Measurement	Interpreted	Management	Unstructured
Ultrasound rec.	Questionnaires	Customer Billing	Scientific papers
Blood pressure recordings	Dictations		Booklets
ECG recordings	Local Diagnostic	Personnel	Information brochures
Spirometric values	Ext Diagnostic	Orderings	
Ergometric values		Facilities documents	
Laboratory values (int & ext)		Workflow related	
Radiological rec. (ext)		Accounting (out-sourced)	

Table 4.1: Data objects identified in the domain of investigation.

Figure 4.1 presents an overview of data type interconnection within the private practice. The interconnections were empirically derived by interviewing personnel and documenting business processes.

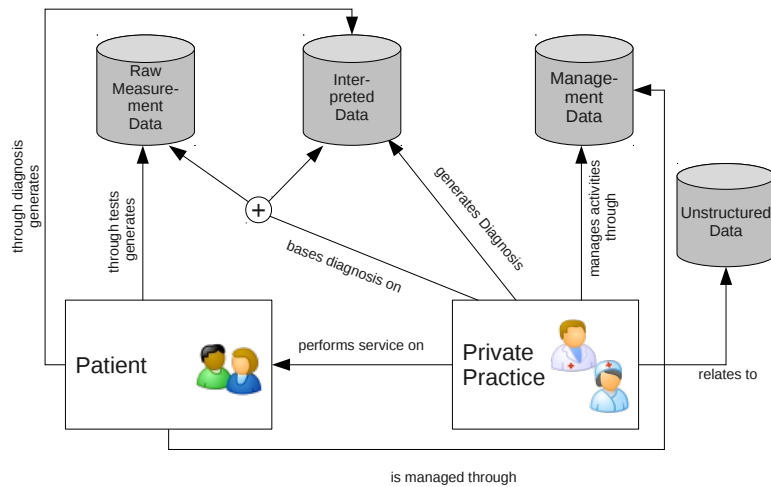


Figure 4.1: Interconnection of Data Types. Data types are passive, being acted upon by subjects.

As can be seen, the interpretation of Raw Measurement Data and existing Interpreted Data (either generated in the own private practice, or retrieved from external sources) by the private practice, leads to new derived Interpreted Data.

Now that data within the practice has been identified, comparted and it's interconnection abstractly defined, qualification and classification can be executed.

4.2 Data Classification

To provide a connection between the identified data types and the upcoming analysis, a transformation into an information layer model is given. The information layer model is part of the enterprise architecture, the analysis method is on one part based on, and allows for embedding into the business processes respectively workflows. Certain data types are integrated into single elements within the information model as to provide a concise overview.

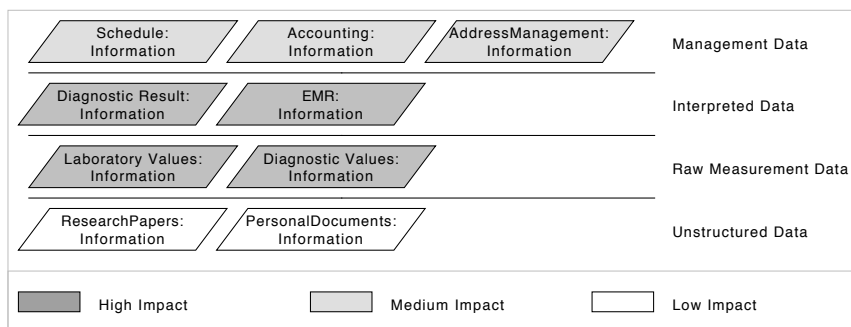


Figure 4.2: The local information layer model. The information entities are colored according to their business impact on loss or degradation.

Figure 4.2 presents this local information layer model. Certain data types, as identified, are integrated into single elements. The mapping from those identified data types to their respective information entity is pragmatic and can be derived from the figure. The **LaboratoryValues: Information** element for example subsumes all data types found within the category of Raw Measurement Values (as identified in table 4.1).

Asset valuation has been carried out, by questioning the practitioner about what information entities would, on loss or degradation, demonstrate the highest impact on business (both financial and non-financial). The qualified statement is presented in figure 4.2 by dif-

ferent pigmentation of information entities. It shows that `Diagnostic Result:Information`, `EMR:Information`, `Laboratory Values:Information` and `Diagnostic Values:Information` constitute the main data assets.

It is worth noting, however, that diagnostic results and diagnostic values are directly produced and consumed by the practitioner. Hence those data assets are not specifically involved in the analysis domain. `Laboratory Value:Information` is generated either by in-house or external laboratories, being more relevant to an analysis of semantic integrity than security per se.

4.3 Modeling the sample workflow

The analysis has to base on a meaningful sample workflow. Selection of the workflow is carried out considering the data asset with the highest BIA and business process rate of occurrence. Classification of data assets according to BIA has already been carried out hence an overview of business processes and their respective occurrence rate has to be created. Figure 4.3 provides a qualitative occurrence rate overview of business processes for the selected use case. The classification has been done by on-site observation during worktimes.

Consolidating this information, a sample workflow integrating `FullCheckup:BusinessProcess` and `EMR:Information` is the focal point of the analysis.

In order to create the required foundation, the necessary models, according to the ProSeCo analysis process, need to be created, and to provide a full overview and the identification of dependencies, the respective layers (or views) have to be modeled first. During this process the intra-layer dependency identification is anticipated, while the inter-layer dependency identification is carried out as a separate process.

4.3.1 Business layer

A full depiction of the business layer consists of the organizational, the business process and the informational view. Business process view and the information layer (view) have already been presented in figures 4.3 and 4.2 respectively.

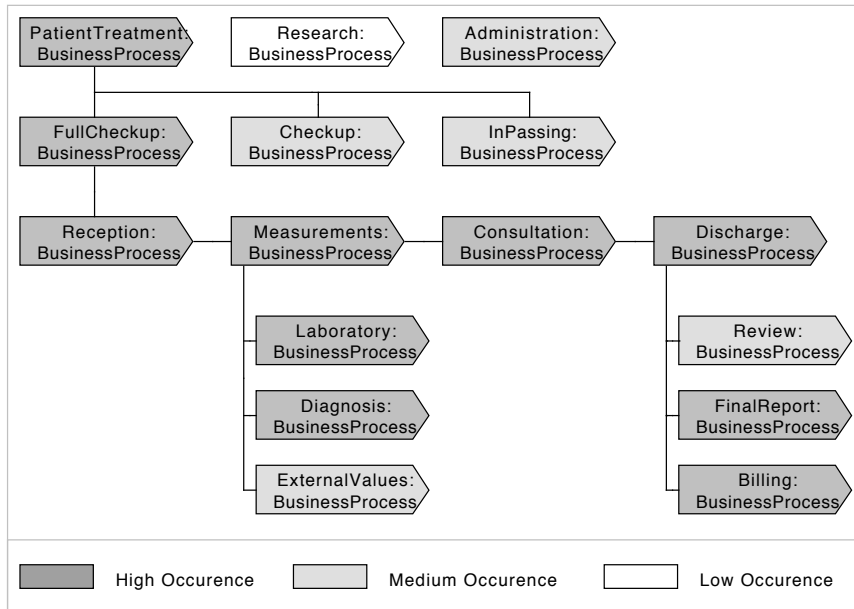


Figure 4.3: Business Processes occurring within the medical practice with a focus on patient treatment and the respective occurrence rates.

To complete the business layer we hence need to include the organizational model, which is presented in figure 4.4. Naturally, due to the case studies domain of investigation, this view is sparsely populated.

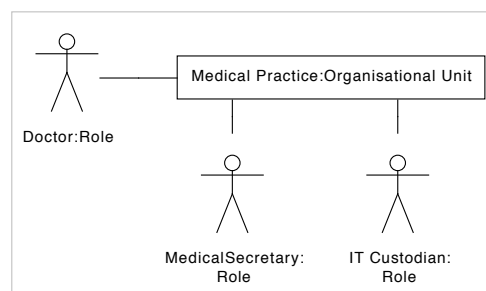


Figure 4.4: Organizational model of the medical practice.

4.3.2 Application layer

To provide a concise application view, the relevant (for the sample workflow) applications (resp. components) are grouped into three categories (they can be seen as rising in generality top down):

- *Domain specific applications*; application and software related to the specific domain of investigation, i.e. the EMR software, software driving lab devices, etc.
- *Generic applications*; applications found in every (office) environment, such as office tools and web browsers
- *Infrastructure components*; components enabling the infrastructure, or all the required software to revive the hardware.

The general assumption is that both domain specific and generic applications require infrastructure components to operate. The application view is provided in figure 4.5. As one can see, the applications and components increase in generality top-down.

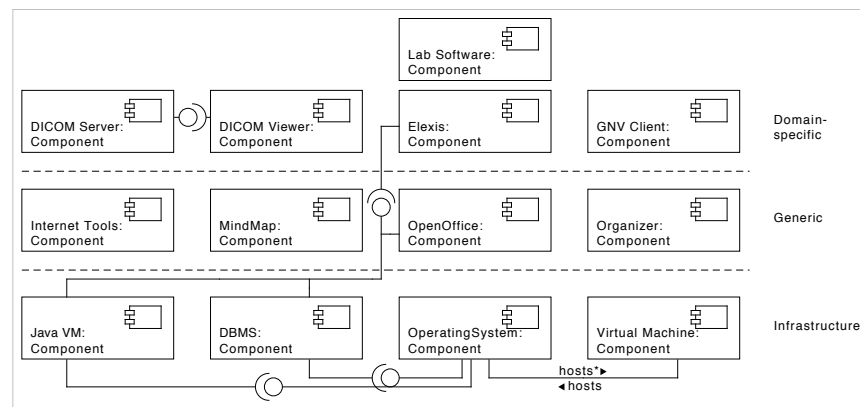


Figure 4.5: (Partly) Component resp. application model of the medical practice. The `hosts*` denotes that it depends on the type of virtualization used, see [DB06].

Anticipating the identify dependencies activity (within the ProSeCo security process), inter-layer-dependency is presented using standard Unified Markup Language (UML) component model schemata. Elexis is in

a require-relationship with a Database Management System (DBMS), a Java Virtual Machine (JVM) and an Office component (currently OpenOffice). Without these required components, Elexis does not provide its set of functionality, demanded by the business process activities it is involved in.

The Digital Imaging and Communications in Medicine (DICOM) related components are specific to ultrasound related activities, they are not included in the analyzed business process, but represented for the sake of presentation.

4.3.3 Software layer

The implementation layer extends the application layer by combining an applications SW-Architecture and an Information entity showing the implemented code. In the selected software component *Elexis*, the `EMR:Information` maps to the Java class `Konsultation.java`, which is tightly related with `Fall.java`.

Before diving into the direct implementation of the respective classes, it is necessary to gain an understanding of the underlying data model and its relationship, that is its rationale.

Figure 4.6 depicts the (use case respective) relevant data model to form an EMR. The rationale cycle is as follows: A *Fall* (case) is a series of *Konsultationen* (Treatments). A *Konsultation* generates a *Diagnose* (Diagnosis), and/or results in an *Eintrag* (Entry). EMR is a fluent concept, as it only expresses data about the patient. So to define the meaning in this use case, we need to relate to a single EMR either by defining it as a single *Fall* or a single *Konsultation*.

It is worth noting at this point, that Elexis defines a standard for data exchange: “Standard für den Austausch medizinischer Daten” (SamDas). In this standard the *Fall* is the element to be exchanged between the health network participants. In Austria the *Austrian Hauptverband der Versicherungen* does mandate the implementation of the *Exportnormdatensatz* [Aus10] in applications that invoice the national social insurance.

To subsequently analyze the respective classes, we need to model a class diagram. Due to the complexity of software, a reduction to the core elements of the EMR is necessary, before continuing deeper modeling. That

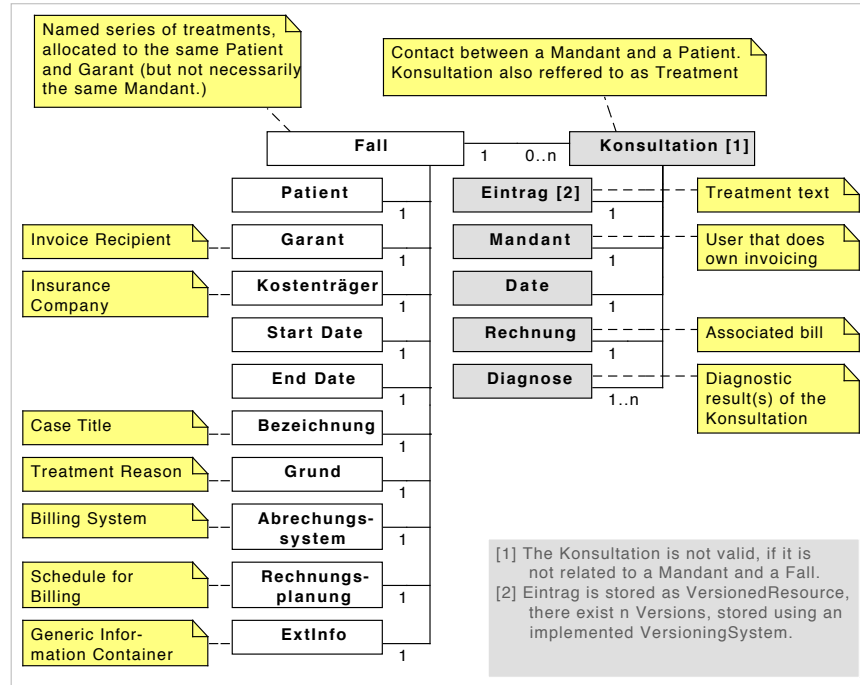


Figure 4.6: Data Model showing the information entities Fall and Konsultation forming the EMR.

is, we reduce the elaborate class analysis on the elements that constitute the information values worth protecting. So we skip, for example, the elements concerning meta information or the insurance company the bill is accounted with. The selection approach is rather pragmatic in this analysis case (due to its academic nature). If, however, a compliance analysis is to be carried out, a source of regulation, defining required protection levels, is [ISO08].

To confine the modeling to the use case, as stated in the white box approach in chapter 3.1.3, three separate parts need to be elaborated: the Elexis Access Control Model, the “surroundings” of `Konsultation.java` and `Fall.java` (i.e. the data model) and the interplay of both. The detailed information shown in the class diagram is reduced as to only include methods that perform CRUD operations on EMR relevant elements. Additionally the class responsible for persisting the information is relevant to include.

Elexis access control system

Figure 4.7 provides a class model overview of the Elexis access control system.

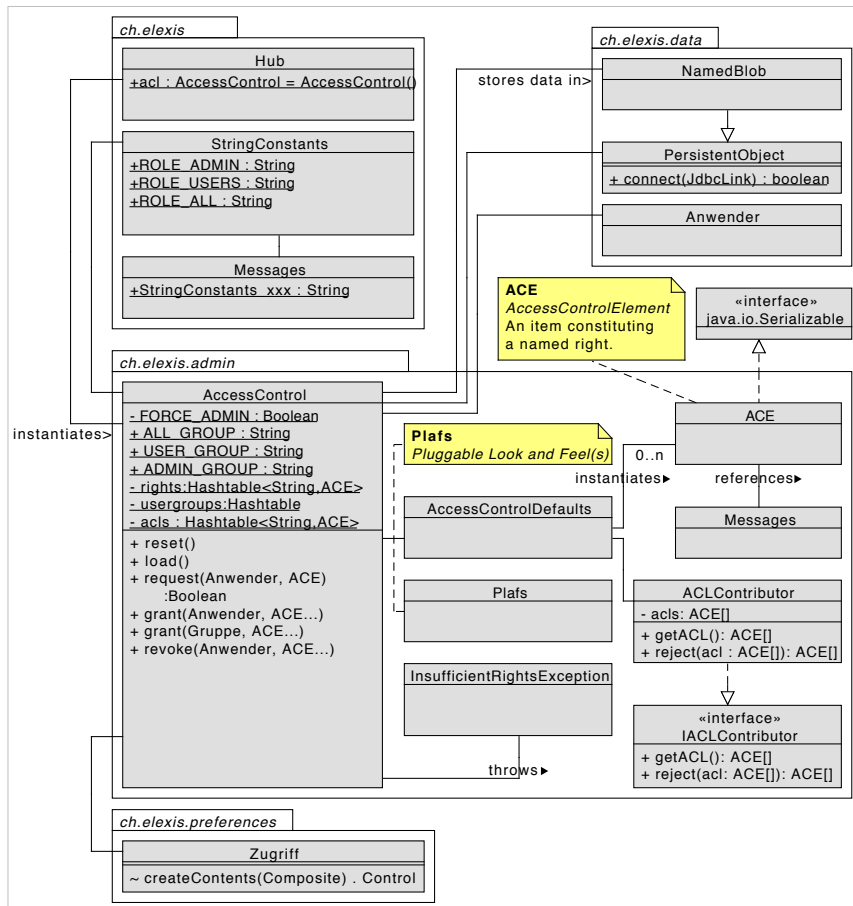


Figure 4.7: Overview of the Elexis access control system.

`ch.elexis.Hub` is the basic entry point of the program, being the plugins Activator class. It instantiates the access control as a constant. The class `ch.elexis.admin.AccessControl` forms the access and rights management system. The rationale of the system is:

- There are groups and users.
- Each user belongs to at least one group.

- From the outset there is a group *Alle* and a user *Jeder*.
- Each right can be conferred to one or more groups and one or more users.
- A user receives all rights either belonging to him, or one of the groups the user belongs to.

A resource requiring access protection has to create an Access Control Element (ACE) for each specific right. Access rights can be granted hierarchically (that is considering an access right A/B/C: If there is no access right for C, a rule for B is searched and so on).

An ACE is an item constituting a named right, they are created in `ch.elexis.admin.ACE.java`. ACEs are collected hierarchically using Access Control List (ACL) collections. An ACE has a parent, an internal name and a (probably localized) external name that will be shown to the user.

The access right elements (`rights`, `acls` and `usergroups`) are loaded (using the `Hub.acl.load()` method) by either one of two methods: `PersistentObject` on set-up of a database connection (within the `connect()` method) or `Zugriff` when creating the contents of the preference page presented within the application (using the `createContents(Composite)` method).

EMR related classes

Now that the basic access control system has been depicted, we concentrate on the code representation of the EMR data model (cf. figure 4.6). An excerpt (considering relevance for the analysis) overview of the included classes, methods and functions is presented in figure 4.8.

The EMR is implemented using the classes `ch.elexis.data.Konsultation` and `ch.elexis.data.Fall`. Both of these classes extend the `PersistentObject` class, which provides the persistence architecture within Elexis.

As can be seen, the persistence architecture includes an inter-component dependency, in the specific case realized using the JDBC API (the Java VM inter-component dependency is not specifically depicted). So we

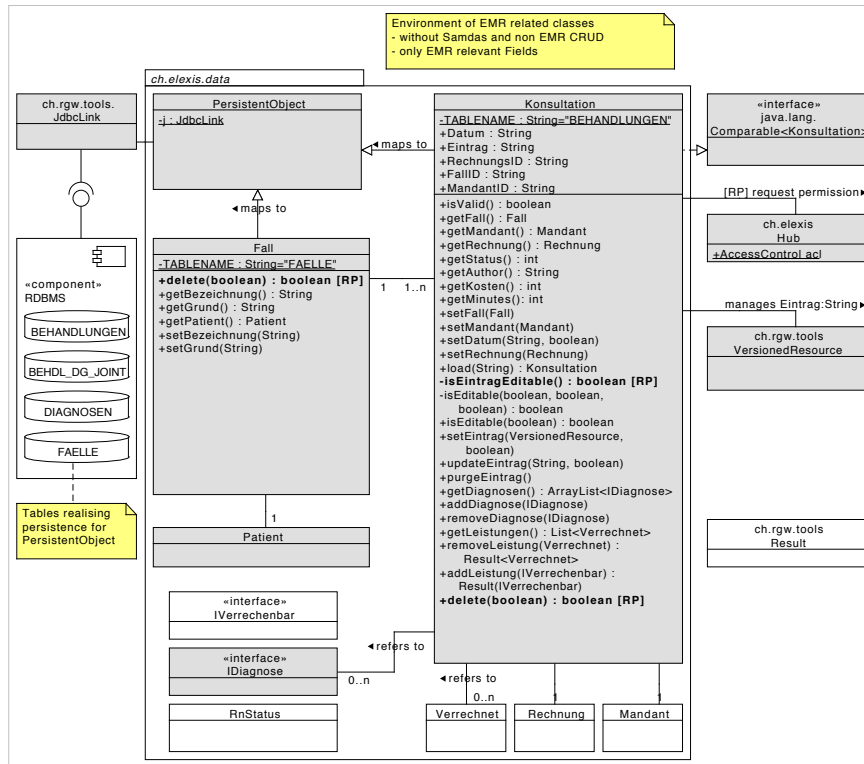


Figure 4.8: Classes realizing the EMR data model, as shown in fig. 4.6. The methods marked with [RP] do currently implement an access control check.

have to keep in mind to give special attention to the characteristics of the shown requires/provides connection during the subsequent analysis. Methods marked with [RP] do request a permission for their respective task to be executed. E.g. the method `isEintragEditable()` in `Konsultationen`, queries for the `AccessControlDefaults.ADMIN_KONS_EDIT_IF_BILLED` right at `Hub.acl`. If there exists an association with the current mandant to this right, access is granted by returning the boolean `true`.

4.3.4 Physical layer

The physical layer is subdivided into the *technical layer*, representing the physically existing hardware devices, and the *physical layer* representing

the real-world location of these devices.

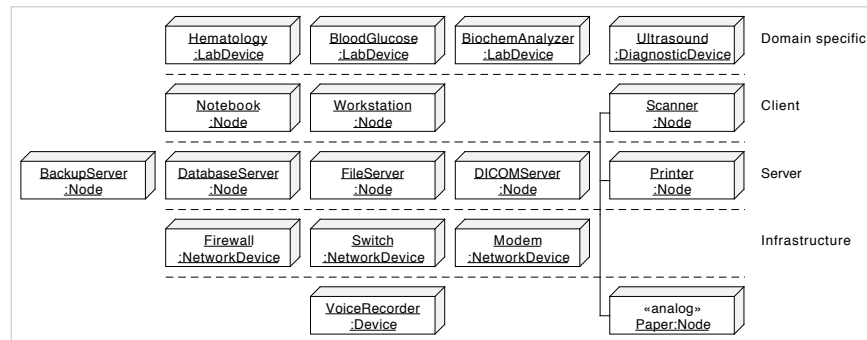


Figure 4.9: Technical Model, resp. overview of physical resources.

Figure 4.9 provides an overview of the systems available within the domain of investigation. The elements, resp. nodes are classified according to their “usage aspect”, that is whether they are being used mainly as clients, servers, domain specific or infrastructure devices. This classification eases understanding and may be used for dependency or relation depiction, as shown before in the other layers. To account for the quality requirements, for example, the relation with `Printer:Node` and `Scanner:Node` to `<<analog>>Paper:Node` is graphically depicted. The physical model with respect to the rooms and buildings is omitted, as it is vacuous in the analysis environment.

4.4 Deriving security and quality objectives

In [Mit08, figure 17, page 65] the following local security objectives and resulting security requirements, concerning the EHR are identified for a general practitioner:

- SO_01: Identification of health network participators
 - SR_28: Unique identification
- SO_02: Authentication of health network participators
 - SR_13: Assurance of identity

- SO_03: Authorization of participators
 - SR_37: Approval of patient
 - SR_38: Prevention of emergency access misuse
- SO_04: System Immunity
 - SR_25: Security management
- SO_05: Integrity of health related data
 - SR_02: Completeness of EHR
 - SR_20: Educated users
 - SR_33: Correct mapping of documents
- SO_06: Non-repudiation
 - SR_14: Legally binding accountability for activities
- SO_07: Security of health related data
 - SR_19: Protection of patient data
 - SR_40: Prevent patient selection by insurance companies
 - SR_43: Diligent handling of EHR
- SO_08: Security assessment
 - SR_11: Compliance to IT security standards
- SO_10: Physical asset protection
 - SR_27: Plan for human malpractice
 - SR_01: Access to system rooms

Considering these objectives, the following quality objectives and requirements could be derived by considering S0_05:

- QO_01: Semantic integrity of health related data
 - QR_01: Referring to SR_20: Domain dependent knowledge as to allow data stewardship
 - QR_02: Error control on analog / digital data conversion

- QR_03: Assurance of semantic correctness of data

For the security analysis the following security objective and requirement is selected for treatment within this use case: S0_07/SR_19. The selection is reduced to a specific requirement, as to focus on a specific aspect within the implementation view.

4.5 The sample workflow inter-layer dependencies

Figure 4.10 presents the inter-layer dependencies of `AquaintWithFacts:Activity` in the `Consultation:BusinessProcess`. As `Doctor:Role` is connected to the business process, it is valid for all activities carried out within it. `AquaintWithFacts:Activity` is related to `EMR:Information` and the two components `DICOM Viewer:Component` and `Elexis:Component`. `Elexis:Component` is depending directly on `Workstation:Node` and transitively (via its require relationship to other components) on `DatabaseServer:Node`.

4.6 Security and quality requirements engineering

In chapter 4.4 the following security and quality objectives (resp. requirements) were selected for consideration in this use-case:

- S0_07 (Security of health related data)/SR_19
- Q0_01 (Semantic integrity of health related data)/QR_01
- Q0_01 (Semantic integrity of health related data)/QR_02
- Q0_01 (Semantic integrity of health related data)/QR_03

As the ProSecO relative treatment of the enterprise architecture resp. the sample workflow is already described within [Mit08], the security analysis is focused on considering these security requirements within the implementation view. A schematic depiction of this within ProSecO is presented in figure 4.11.

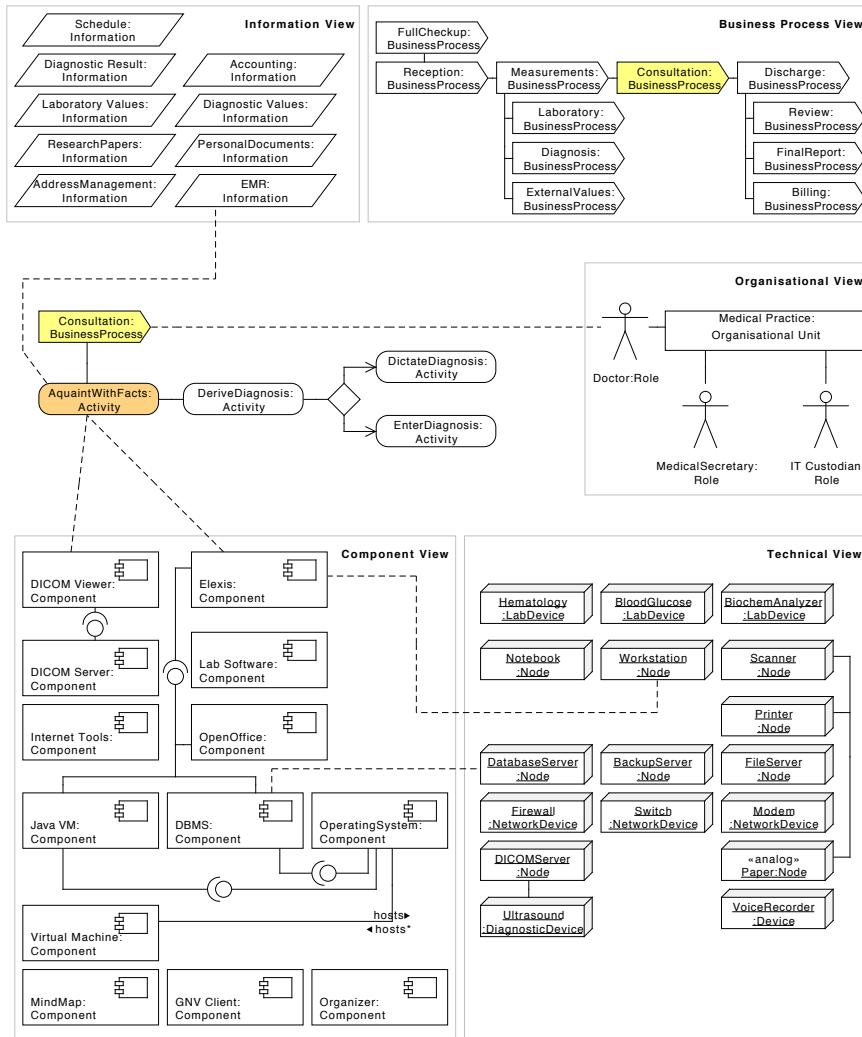


Figure 4.10: ProSecO architecture inter-layer dependency. The implementation view is not shown.

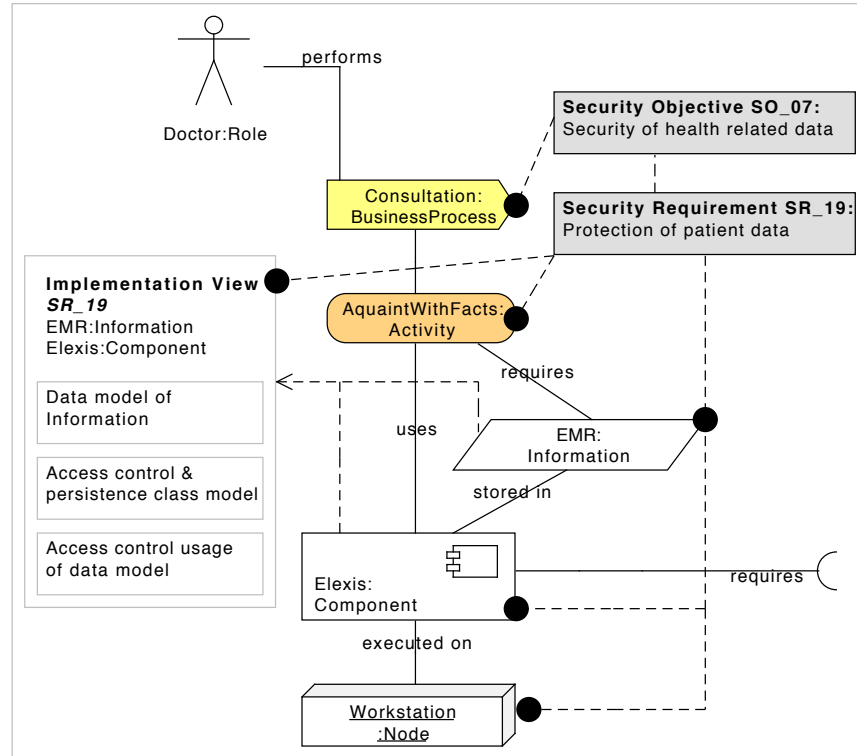


Figure 4.11: Applying the security objective and security requirements to `EMR:Information` and `Elexis:Component` inherits the security requirements to the implementation view.

The quality objectives and requirements, however, are still to be considered within the enterprise architecture view. The sample workflow selected so far, does provide a single point of contact for a quality threat. That is, if the practitioner chooses to dictate the diagnosis for a patient onto a voice recorder (cf. `DictateDiagnosis:Activity` and `VoiceRecorder:Device` in figure 4.10).

The quality case, as presented in figure 4.12 is considered for the purpose of demonstration, and not included in the subsequent analysis.

There is hardly any points to mitigate these (quality) threats at an infrastructure level, except for the automation of data conversion. The probability of such a mitigation to be realized in the depicted example is very low, as the automated and meaningful recognition of human

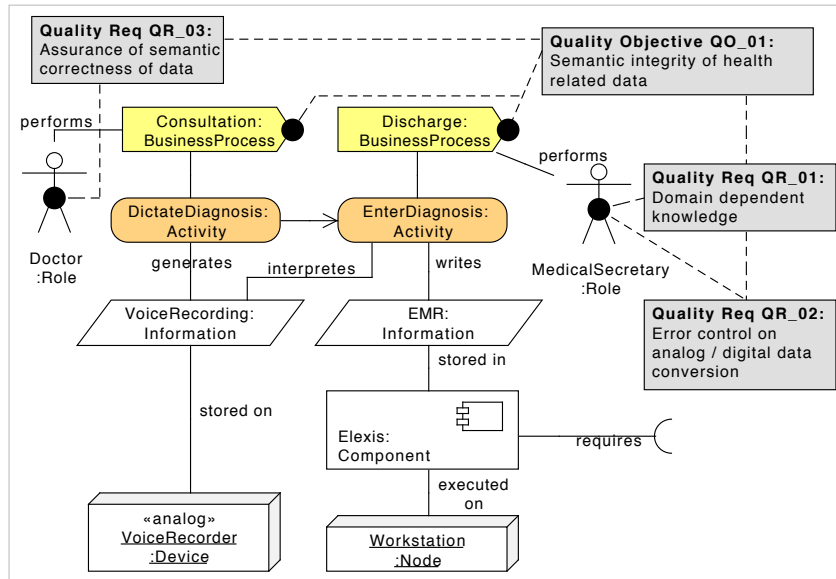


Figure 4.12: A sample demonstration of quality objectives and quality requirements to consider threats to semantic integrity.

voice is still an own area of research. If however the example depicts an analog conversion of laboratory values into the system, the possibility of automatic data conversion may be given. In the course of this master thesis, such an automatic conversion was programmed by the author to eliminate quality threats on laboratory value entries [DW10].

4.7 Threat and risk analysis

Threat and risk analysis within this use case is focused on the handling of `EMR:Information` within `Elexis:Component`. That is, the focus of investigation lies within the respective implementation view, and its compliance to `SO_07` (Security of health related data)/`SR_19` (Protection of patient data).

We follow the steps described by (simplified) white-box security testing in chapter 3.1.2.

4.7.1 Understand the rationale

During the process of modeling the relevant classes and the access control system (cf. figure 4.7 and figure 4.8), an idea of possible security problems is generated by the analyst. This results in some questions and remark points, which are noted for further consideration during the mitigation of risks (cf. chapter 4.8).

4.7.2 Compare rationale with design principles

First off the security mechanism of the application itself need to be analyzed before diving into the application of the security mechanism by the implementation relevant data.

This analysis amounts to comparing whether certain design principles are upheld, as described in chapter 3.1.2, the respective report is presented in table 4.2.

4.7.3 Model the relevant aspects

The relevant aspects have already been modeled during the respective creation of the layers in chapter 4.3. To grasp, however, all the analysis relevant aspects within the model, is an iterative process. So during the execution of the analysis some modifications to the models presented in figure 4.7 and 4.8 were done. These modifications were directly stored back into the model, so the figures seen there are the final product of this iterative process.

4.7.4 Compare model with patterns and rules

The comparison is split into two logical parts, the comparison of the model with design patterns provided in literature and the analysis of the implementation itself.

Design and security patterns

Several security patterns exist, lending itself for comparison with in this case.

During the modeling, in the previous step, analogies to the following patterns, taken from [SFBH⁺06] were detected (the numbers in parenthesis

Design Principle	Compliance	Justification
Least Privilege	Yes	Supported by the access control system, respective implementation is however dependent upon the single parts implementing a respective query.
Fail-Safe Defaults	No	The methods of information elements classes do either query for a decision on access rights on certain tasks, or simply continue with the required work (cf. figure 4.8 and the [RP] remarked methods).
Economy of Mechanism	Yes	The security mechanism is clear cut and easy to understand. It is straightforward to implement access queries into methods, and the respective codebase is clearly arranged.
Complete Mediation	No	Access rights are stored in a <code>NamedBlob</code> which is a subclass of <code>PersistentObject</code> implementing a cache. Although there exist synchronization methods, a possible unsynchronized state can not be excluded.
Open Design	Yes	Open source code.
Separation of Privilege	No	Access to a certain aspect is dependent upon an existing access control element only.
Least Common Mechanism	Yes	Access to resources is handled implicitly using the <code>PersistentObject</code> class, all persistent elements are a subclass to it.
Psychological Acceptability	Yes	The user only has to provide username and password during login, access right checks are executed invisible to the user.

Table 4.2: Compliance of Elexis access control rationale resp. implementation to design principles.

denote the respective page number in the book):

- **Access Control Models**
 - AUTHORIZATION (245)
 - ROLE-BASED ACCESS CONTROL (249) (RBAC)
 - REFERENCE MONITOR (256)
 - ROLE RIGHTS DEFINITION (259)

- **System Access Control Architecture**
 - SINGLE ACCESS POINT (279)
 - CHECK POINT (287)
 - SECURITY SESSION (297)
 - FULL ACCESS WITH ERRORS (305) or LIMITED ACCESS (312)

A comparison of these security patterns with the Elexis related application resp. implementation is presented in table 4.3*a*, 4.3*b* and 4.3*c*. Here the original pattern as depicted in [SFBH⁺06] is compared to the actual implementation within Elexis 2.0 and a comment on the respective implementation is given.

Implementation rules

The implementation rule analysis is dependent on the programming language. As already mentioned, Elexis is implemented using Java, hence respective guides such as [Ora10] and [Whe03, chapter 10.6] are a prime source for language specific implementation and security rules.

As the source code is fully available, it is also possible to apply automated analysis tools on the code. [RAF04] provides a comparison of bug finding tools for Java.

Exemplarily FindBugs [HP04] and PMD [C⁺10] were applied. The respective result can be seen in table 4.4.

FindBugs and PMD differ in their respective methods to reveal bugs and warn about certain implementation mistakes. FindBugs analyzes the bytecode using syntax and dataflow analysis, PMD analyzes the source code using syntax parsing. FindBugs classifies bugs according to

Pattern	Pattern Structure	Implementation in Elexis	Comment
<p><i>Access Control:</i> AUTHOR-IZATION</p>			<p>The pattern is realized by ACEs structured in ACLs, as described in chapter 4.3.3. The ProtectionObjects, however, are responsible for checking these access rights for themselves, see REFERENCE MONITOR on this.</p>
<p><i>Access Control:</i> ROLE-BASED ACCESS CONTROL</p>			<p>Groups (Roles) are represented as strings consisting of comma separated values of group memberships of a user. All available groups are stored in a global config file. Groups, however, do not represent functional roles within the enterprise, but only group a set of access rights.</p>
<p><i>Access Control:</i> REFERENCE MONITOR</p>			<p>The requests are generated by methods of objects requesting access rights, by simply calling for a specific ACE to exist. There is no differentiation between an abstract and concrete reference monitor.</p>

Table 4.3 a: Comparison of security patterns and Elexis implementation.

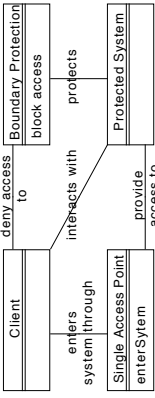
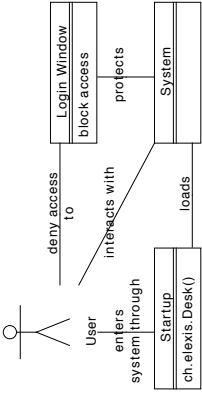
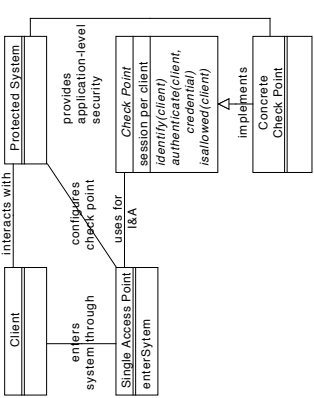
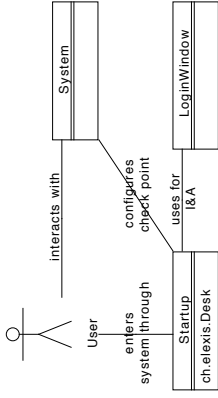
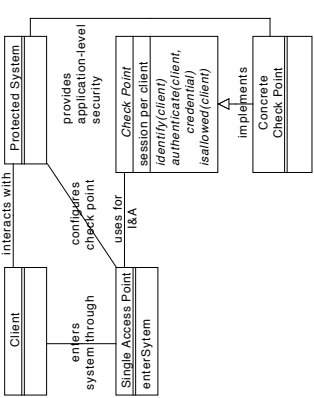
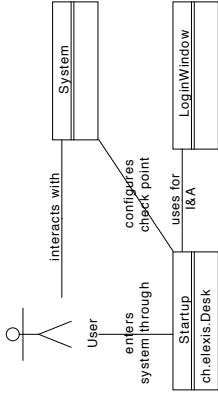
Pattern	Pattern Structure	Implementation in Elexis	Comment
<p><i>Access Control:</i> ROLE RIGHTS DEFINITION</p>	<p>Define the use cases of the system. Use case defines the interaction of actors with the system, we can interpret actors as roles. Roles that appear in a use case must be authorized for all the operations initiated by the role, or role can not perform its functions.</p>  <pre> graph TD Actor[Client] -- "enters system through Single Access Point enterSystem" --> UC1[Boundary Protection block access] Actor -- "enters system through Single Access Point enterSystem" --> UC2[Protected System] UC1 -- "provide access to" --> UC2 UC1 -- "interacts with" --> UC2 UC1 -- "protects" --> UC2 </pre>	<p>Access rights are created on a need to have basis. These access rights can be associated to a group and a user. Use cases are not considered, as access rights are freely associated according to the needs of the respective user.</p>  <pre> graph TD Actor[User] -- "enters system through Startup ch.elexis.Desk()" --> UC1[Login Window block access] Actor -- "enters system through Startup ch.elexis.Desk()" --> UC2[System] UC1 -- "deny access to" --> UC2 UC1 -- "interacts with" --> UC2 UC1 -- "protects" --> UC2 </pre>	<p>Often there is no clear differentiation of roles within a practice (concerning specific tasks within the system), hence a free allocation of access rights fits the environment.</p>
<p><i>System access architecture:</i> SINGLE ACCESS POINT</p>	 <pre> graph TD Actor[Client] -- "enters system through Single Access Point enterSystem" --> UC1[Protected System] Actor -- "enters system through Single Access Point enterSystem" --> UC2[Check Point session per client, identify(client), authenticate(client), validate(client), isAllowed(client)] UC1 -- "interacts with" --> UC2 UC1 -- "provides application-level security" --> UC2 UC1 -- "configures check point" --> UC2 UC1 -- "uses for I&A" --> UC2 UC2 -- "implements" --> UC3[Concrete Check Point] </pre>	 <pre> graph TD Actor[User] -- "enters system through Startup ch.elexis.Desk" --> UC1[System] Actor -- "enters system through Startup ch.elexis.Desk" --> UC2[LoginWindow] UC1 -- "interacts with" --> UC2 UC1 -- "configures check point" --> UC2 UC1 -- "uses for I&A" --> UC2 </pre>	<p>As the application is standalone, the single access point is given at the startup of the system which enforces a login window. After successful authentication the user is allowed to interact with the system. On unsuccessful authentication it is possible to end application execution.</p>
<p><i>System access architecture:</i> CHECK POINT</p>	 <pre> graph TD Actor[Client] -- "enters system through Single Access Point enterSystem" --> UC1[Protected System] Actor -- "enters system through Single Access Point enterSystem" --> UC2[Check Point session per client, identify(client), authenticate(client), validate(client), isAllowed(client)] UC1 -- "interacts with" --> UC2 UC1 -- "provides application-level security" --> UC2 UC1 -- "configures check point" --> UC2 UC1 -- "uses for I&A" --> UC2 UC2 -- "implements" --> UC3[Concrete Check Point] </pre>	 <pre> graph TD Actor[User] -- "enters system through Startup ch.elexis.Desk" --> UC1[System] Actor -- "enters system through Startup ch.elexis.Desk" --> UC2[LoginWindow] UC1 -- "interacts with" --> UC2 UC1 -- "configures check point" --> UC2 UC1 -- "uses for I&A" --> UC2 </pre>	<p>This is almost identical to the SINGLE ACCESS POINT pattern, as there is no direct differentiation within a standalone application. The Login Window provides the single check point of the system, and subsequent environment variables are set.</p>

Table 4.3b: Comparison of security patterns and Elexis implementation.

Pattern	Pattern Structure	Implementation in Elexis	Comment
<p><i>System access control architecture:</i> SECURITY SESSION</p>			<p>There exists only one session at a given time, as the user logs in into the system and the respective variables are set denoting a specific user identity. Signing off is equal to exiting the program. The security session is represented by the user identity related environment variables.</p>
<p><i>System access control architecture:</i> FULL ACCESS WITH ERRORS</p>			<p>Elexis applies this pattern for several operations as setting user and group rights. All possibly available options are shown, where options not available with the access rights of the current user are grayed out. If a specific task where the user does not have sufficient rights for is implicitly called, either no result is returned, or an explanatory message.</p>
<p><i>System access control architecture:</i> LIMITED ACCESS</p>		<p>empty</p>	<p>This pattern is not currently used in the core of Elexis, it is however presented for the sake of completeness, as generally either FULL ACCESS WITH ERRORS or LIMITED ACCESS is selected as a strategy. Possibly plugins not considered in this case study will employ this specific pattern. It is however encouraged, to increase usability, to select and employ only one strategy.</p>

Table 4.3c: Comparison of security patterns and Elexis implementation.

Target	FindBugs	PMD
Elexis 2.0	16 High / 87 Normal / 7 Low Priority	100 Errors / 9658 Warnings
Fall		3 E / 54 W
Konsultation		5 E / 74 W
PersistentObject	1 H / 1 N	10 E / 161 W
AccessControl	1 H	1 E / 40 W
ACE		10 W

Table 4.4: Automated analysis of Elexis 2.0 implementation using FindBugs and PMD. Selected classes, relevant for the case study are additionally indicated.

priority, where a high priority denotes a definite bug and lower levels indicate the possibility of a bug existing.

Table 4.5 presents the result of the FindBugs analysis on the complete Elexis 2.0 project. The identified bugs are ordered by their respective number of occurrences.

The topmost error “Field isn’t final and can’t be protected from malicious code”, for example, refers to the bug that a mutable static field could be changed by malicious code or by accident from another package. This is due to the fact that Java allows to override methods of the superclass an attack class is extended of, if the respective method is not marked final in the superclass. Or as a statement by John Nightingale cited in [Whe03] declares: *... the big thing with Java programming is minding your inheritances. If you inherit methods from parents, interfaces, or parents’ interfaces, you risk opening doors to your code.*

An overview and full description of all bugs identified by FindBugs is available at [Fin10].

Table 4.6 presents the errors and the error description of violations found by PMD within the case study relevant classes. The “Avoid Reassigning Parameters” violation creates the highest violation count (6 times), and as the note in [C⁺10, Current Ruleset] reveals, doing so is a questionable practice. Method parameters should be final, as to not introduce any unwanted side effects.

FindBugs identified bug	#	Category
Field isn't final and can't be protected from malicious code	47	MAL_CODE
May expose internal representation by incorporating reference to mutable object	14	MAL_CODE
Field should be package protected	12	MAL_CODE
Possible null pointer dereference	10	COR
Field isn't final but should be	9	MAL_CODE
May expose internal representation by returning reference to mutable object	5	MAL_CODE
Public static method may expose internal representation by returning array	3	MAL_CODE
Class defines field that masks a superclass field	1	COR
equals() used to compare array and nonarray	1	COR
Field is a mutable array	1	MAL_CODE
Impossible downcast of toArray() result	1	COR
Method ignores return value	1	COR
No relationship between generic parameter and method argument	1	COR
Null pointer dereference	1	COR
Nullcheck of value previously dereferenced	1	COR
Possible null pointer dereference in method on exception path	1	COR
Unwritten field	1	COR
Very confusing method names	1	COR

Table 4.5: FindBugs result for Elexis 2.0 ordered by number of occurrences. FindBugs was configured to report Malicious Code Vulnerability (MAL_CODE), Security and Correctness (COR) issues.

Class	PMD Error [#]	[C ⁺ 10, Current Ruleset] Note
Fall	Avoid Reassigning Parameters (ARP) [1]	Reassigning values to parameters is a questionable practice. Use a temporary local variable instead.
	Constructor Calls Overridable Method [1]	Calling overridable methods during construction poses a risk of invoking methods on an incompletely constructed object and can be difficult to discern (...)
	Method Naming Conventions (MNC) [1]	Method names should always begin with a lower case character, and should not contain underscores.
Konsultation	Avoid Using Volatile [1]	Use of the keyword 'volatile' is general used to fine tune a Java application, and therefore, requires a good expertise of the Java Memory Model. (...)
	ARP [3]	see above
	MNC [1]	see above
Persistent Object	Empty Method in Abstract Class Should be Abstract [3])	An empty method in an abstract class should be abstract instead, as developer may rely on this empty implementation rather than code the appropriate one.
	ARP [2]	see above
	Integer Instantiation [1]	In JDK 1.5, calling new Integer() causes memory allocation. Integer.valueOf() is more memory friendly.
	Variable Naming Conventions (VNC) [3]	Final variables should be fully capitalized and non-final variables should not include underscores.
	Return Empty Array Rather Than Null [1]	For any method that returns an array, it's a better behavior to return an empty array rather than a null reference.
Access-Control	VNC [1]	see above
ACE	None	

Table 4.6: PMD results for selected classes of table 4.4. Only errors are presented. Warnings resp. violations with lower level are omitted.

4.8 Mitigation

As the focus of this thesis lays upon the analysis process, mitigation measurements and their execution is presented in terms of the required continuation steps. The focus here lies on the implementation view respective mitigation procedures.

The overall security risks identified during the analysis process, are subject to standard mitigation procedures as presented e.g. in the german IT-Grundschutz Kataloge [Bun10] or derived from best-practice and governance libraries like ITIL [Off10] and COBIT [Inf10].

The *IT-Grundschutz Manahmenkatalog* and the *IT-Grundschutz Bausteine* for example offer special sections on *Standardsoftware (B1.10)* (common software) and *Software-Abnahme- und Freigabe-Verfahren (M2.62)* (software approval procedures) where the preceding analysis provides valuable input information.

Certain aspects, however, need to be directly clarified with the respective software developer and care has to be taken for the concerns to be considered.

As already mentioned, during the analysis process the analyst identifies certain questions which need to be clarified with the respective programmer. These identified questions are dependent upon the analysts experience and her ability to identify *code smells*, especially concerning security related problems.

The following questions were identified by the analyst during the analysis process, and will subsequently have to be addressed by the software developer:

- Is the JDBC link encrypted?
- Is the database connection string readable?
- Is it possible for a plugin to circumvent the access control?
- Is it possible for a plugin to maliciously modify the access control?
- Is it possible to write a plugin that fakes a user and/or generates administrator rights?
- Can a malicious file trick an importer into some misuse case?

- Is it possible to access private elements in the field through reflection?
- Can a class be instantiated without a proper identification (as Mandant)?
- `Konsultation.purgeEintrag()` is protected only by user interface call access control element check! What if a plugin calls this? Is there any protection?
- Does the program enforce the usage of secure passwords?
- Does the implementation of role-based access control and the implementation of the reference monitor follow best-practices resp. do they conform to a proven security pattern?
- Shouldn't the single persistent objects be responsible for their own protection, i. e. on CRUD methods, the object should check the permission?
- Can third-party plugins be forced to implement specific requests for access permission by the core system?
- Is there a sufficient separation of the core system to the plugins sufficient to cope with sudden "crashes" of plugins without compromising the stability of the whole program?
- Currently the database system does not possess any logic of the hosted data, hence it is not able to care for self-validity. Is it feasible to change this situation?
- Access rights are only taken care of by the system, once a connection to the database is given the respective link has full database access; is a separation of privilege in this regard feasible?

The software developer then has to create a statement on the respective questions and a mitigation or (if applicable) a fix to a (possibly) disclosed vulnerability. Additionally the software developer may be encouraged in applying the respective design patterns to anticipate flaws.

4.9 Report

The analysis procedure generates several artefacts and models. These can be separated according to whether they are of descriptive kind or analysis results. Descriptive artefacts provide information on the internal workings of the application and present knowledge on their relations and connections. Analysis results are generated out of evaluating the descriptive artefacts against a certain set of best-practices, rules or samples that have been proven to be correct.

Both the descriptive and result artefacts are delivered to the initiator of the case study, who could use the information for a decision on whether to deploy the specific software in the current state or request reworkings. The results could also be applied by management to estimate a certain maturity level considering the security awareness of both the enterprise and software architecture.

It is worth noting at this point, that several software development projects already integrate the usage of code analysis tools like the ones presented into their overall Software Development Lifecycle (SDLC), which resulted in higher stability in the generated code, cf. eg. [APM⁺07].

Generated artefacts

In the treated case study the following *descriptive artefacts* are created:

- Models representing the single layers and their respective intra-dependency.
- Enterprise Architecture (EA) model presenting the inter-dependencies of the respective layers.
- Data model of the information to be analysed within a specific component.
- Class diagrams of the implementation view relevant parts (cf. chapter 3.1.3).

The analysis process itself generates the following *results*:

- Sample depictions of security objectives and requirements adhered to the elements of the EA model.

- Comparison of the implementation against design principles
- Comparison of the implementation against identified related security patterns
- Comparison of the implementation against implementation rules
- Automated results of the respective software analysis tools

Chapter 5

Summary

This chapter provides a summary on the key results, followed by a reflection on the task and the authors personal opinion.

5.1 Achievement of objectives

The task as formulated was an integrated security analysis of the open source HIS Elexis. This is realized by identifying the EMR as the most-relevant data-asset, and a more profound analysis of its representation within the Elexis component.

As a result to this task ProSecO is modified in the aspect as to consider the implementation of a specific information entity within a component, and to employ a white-box analysis approach. Additionally the possible threat of loss of semantic integrity is identified, and comments on specific ProSecO application problems in local domains are pointed out.

The result is a profound depiction of the application, its inner workings concerning security measures and their relation to the identified data asset and its application of general security patterns and identification of implementation flaws. This provides a ready overview and allows for educated decisions and risk mitigations.

So the following modifications to the original analysis method ProSecO are realized:

- Consideration of the **implementation layer**, merging an `:Information` and a `:Component` element.

- Identification of threats to **semantic integrity** and a scheme for integrating these threats into the ProSecO analysis process.
- Suggestion of **special aspects** to consider when employing the ProSecO analysis method **in the local domain**.

The necessary meta-model modification to include these into the ProSecO process were also presented. Although the modified ProSecO process was focused on a case study in the health domain, the approach should be readily applicable to any other domain where this kind of analysis is required.

5.2 Reflection on the analysis method

ProSecO features model-based analysis of security risks within an enterprise. The modifications to ProSecO presented within this paper are aimed to fit into this specific area of application, that is it is assumed that there is a limitation on monetary, human and time resources. It aims to provide a good overview of the current situation and an educated insight into the security of a system with respect to a specific data asset.

Due to the consideration of application internal workings, an analyst offering evaluation of security should have a well-founded knowledge about the language of the application to be examined.

This thesis treated the topic of white-box analysis with the justification of the source code being available. The method, however, is not limited to the source code being available, as there also exist black- and gray-box approaches, allowing limited code testing by focusing on specific misuse cases and attacks.

The consideration of quality threats leads to a growing awareness to this kind of risks within an enterprise. Although quality threats will mostly be mitigated through staff education (both technical and domain-specific), there exist certain parts where additional applications may practically remove the specific threat.

5.3 Closing

It is my personal opinion that the analysis of all aspects of an information system (security, functional, non-functional, quality, . . .) should be carried out in a data centric way. That is, although there are lots and lots of technologies included in a modern enterprise computer network it has to be kept in mind that all this technology only serves the purpose of handling and computing data. If this data is properly identified, classified and subsequently treated upon by its classification, several security incidents and fatal mistakes are never prone to happen. This will become even more important when cloud-computing becomes the IT outsourcing preferred solution.

List of abbreviations

ACE	Access Control Element
ACL	Access Control List
BIA	Business Impact Analysis
BSO	Business Security Objective
CRUD	Create, Read, Update or Delete
CEO	Chief Executing Officer
DBMS	Database Management System
DICOM	Digital Imaging and Communications in Medicine
DG	Data Governance
DQ	Data Quality
DSG	Datenschutzgesetz
DLM	Data Lifecycle Management
EA	Enterprise Architecture
EMR	Electronic Medical Record
EHR	Electronic Health Record
HIS	Health Information System
JDBC	Java Database Connectivity
JVM	Java Virtual Machine
MDA	Model Driven Architecture
MDM	Master Data Management
ProSecO	A Process Model for Security Engineering
RCP	Rich Client Platform
SDLC	Software Development Lifecycle
UML	Unified Markup Language

Bibliography

- [AD02] C. J. Alberts and A. Dorofee: *Managing Information Security Risks: The Octave Approach*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.
- [APM⁺07] N. Ayewah, W. Pugh, J. D. Morgenthaler, J. Penix and Y. Zhou: *Using FindBugs on production software, OOP-SLA '07: Companion to the 22nd ACM SIGPLAN conference on Object-oriented programming systems and applications companion*, ACM, New York, NY, USA, 2007, pages 805–806.
- [Aus10] Austrian Medical Association: *Export-Normdatensatz (ENDS)*, <http://www.sozialversicherung.at>, 2008 (accessed July 8, 2010).
- [Bel05] D. E. Bell: *Looking Back at the Bell-La Padula Model, ACSAC '05: Proceedings of the 21st Annual Computer Security Applications Conference*, IEEE Computer Society, Washington, DC, USA, 2005, pages 337–351.
- [BHLOW08] R. Breu, M. Hafner, F. Innerhofer-Oberperfler and F. Wozak: *Model-Driven Security Engineering of Service Oriented Systems*, Kaschek et al. [KKSF08], 2008, pages 59–71.
- [BIOY08] R. Breu, F. Innerhofer-Oberperfler and A. Yautsiukhin: *Quantitative Assessment of Enterprise Security System, Availability, Reliability and Security, International Conference on*, volume 0, (2008), pages 921–928.
- [Bis03] M. Bishop: *Computer Security – Art and Science*, Addison Wesley Professional, 2003.
- [BM03] E. Baniassad and G. C. Murphy: *Design Pattern Rationale Graphs: Linking Design to Source*, In *IEEE 25th International Conference on Software Engineering*, 2003, pages 352–362.

BIBLIOGRAPHY

- [BMIO⁺10] R. Breu, M. Memon, F. Innerhofer-Oberperfler, M. Weitlaner, M. Breu, M. Hafner, R. Scandariato, K. Yskout, K. Buyens, B. Fontan, F. Paci and E. Chiarani: *D2.1 – An architectural blueprint and a software development process for security-critical lifelong systems*, Technical report, 7th Framework Programm European Union – Security Engineering for Lifelong Evolvable Systems, 2010.
- [Bun10] Bundesamt für Sicherheit in der Informationstechnik: *IT-Grundschutz-Kataloge*, <http://www.bsi.bund.de>, 2010 (accessed July 8, 2010).
- [But09] D. Butler: *Oracle Master Data Management: Executive Overview*, Oracle White Paper, Oracle Corporation, 2009.
- [C⁺10] T. Copeland et al.: *PMD – Don't shoot the messenger*, <http://pmd.sourceforge.net>, 2010 (accessed July 7, 2010).
- [DB06] M. Descher and M. Baumgartner: *Rapid and Minimally Invasive Deployment of Grid Middleware by means of Native Virtualization*, J. Volkert, T. Fahringer, D. Kranzlmüller and W. Schreiner (editors), *2nd Austrian Grid Symposium*, Austrian Computer Society, 2006, pages 238 – 248.
- [dBHL⁺07] F. den Braber, I. Hogganvik, M. S. Lund, K. Stolen and F. Vraalsen: *Model-based security analysis in seven steps – A guided tour to the CORAS method*, BT Technology Journal, volume 25(1), (2007), pages 101–117.
- [Des10] M. Descher: *Data Governance: Introduction and Overview*, 2010, Seminar Paper, Leopold-Franzens-University Innsbruck, Institute for Computer Science.
- [DW10] M. Descher and T. Wolber: *Datenaustausch Geräteplugin: Roche Cobas Mira*, <https://elexis-addons.googlecode.com/svn/trunk/elexis-connect-cobasmira>, 2010.
- [EM06] R. Etges and K. McNeil: *Understanding Data Classification Based on Business and Security Requirements*, ISACA JOnline, volume 5.
- [Fin10] FindBugs: *FindBugs Bug Descriptions*, <http://findbugs.sourceforge.net/bugDescriptions.html>, 2010 (accessed July 7, 2010).

- [HB08] M. Hafner and R. Breu: *Security Engineering for Service-Oriented Architectures*, Springer Publishing Company, Incorporated, 2008.
- [HP04] D. Hovemeyer and W. Pugh: *Finding bugs is easy, OOP-SLA '04: Companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications*, ACM, New York, NY, USA, 2004, pages 132–136.
- [Inf10] Information Systems Audit and Control Association (ISACA): *Control Objectives for Information and Related Technology*, <http://www.isaca.org>, 2010 (accessed July 8, 2010).
- [IOB06a] F. Innerhofer-Oberperfler and R. Breu: *IT-Sicherheitsmanagement auf Basis eines Unternehmensmodells*, DACH Security, Düsseldorf, Germany, 2006.
- [IOB06b] F. Innerhofer-Oberperfler and R. Breu: *Using an Enterprise Architecture for IT Risk Management*, ISSAC06: Proc. Information Security South Africa Conference, 2006.
- [IOMHB09] F. Innerhofer-Oberperfler, M. Mitterer, M. Hafner and R. Breu: *Web Services Security Development and Architecture: Theoretical and Practical Issues*, chapter Security Analysis of Service Oriented Systems- A Methodical Approach and Case Study, IGI Global, 2009, in Press.
- [ISO08] ISO 27799:2008: *Health informatics – Information security management in health using ISO/IEC 27002 (ISO 27799:2008)*, Austrian Standards Institute, Vienna, Austria, 2008.
- [Jah01] D. Jahnel: *Datensicherheitsmaßnahmen nach dem DSG 2000*, GI Jahrestagung (2), 2001, pages 1025–1027.
- [KKSF08] R. Kaschek, C. Kop, C. Steinberger and G. Fliedl (editors): *Information Systems and e-Business Technologies, 2nd International United Information Systems Conference, UNISCON 2008, Klagenfurt, Austria, April 22-25, 2008, Proceedings*, volume 5 of *Lecture Notes in Business Information Processing*, Springer, 2008.
- [MBEH09] M. Mosley, M. Brackett, S. Earley and D. Henderson: *The DAMA Guide to The Data Management Body of*

BIBLIOGRAPHY

- Knowledge (DAMA-DMBOK Guide)*, Technics Publications LLC, Bradley Beach, NJ, USA, first edition, 2009.
- [Mil10] K. Miller: *What is business impact analysis?*, <http://searchstorage.techtarget.com>, 2005 (accessed July 7, 2010).
- [Mit08] M. Mitterer: *Sicherheitsanalyse von serviceorientierten Architekturen mit einer Fallstudie im Gesundheitsbereich*, Master's thesis, University of Innsbruck, Innsbruck, Austria, 2008.
- [MMM⁺01] A. Moulton, A. Moulton, S. E. Madnick, S. E. Madnick, M. D. Siegel and M. D. Siegel: *Cross-Organizational Data Quality and Semantic Integrity: Learning and Reasoning about Data Semantics with Context Interchange, Mediation, Proceedings of the Americans Conference on Information Systems (AMCIS)*, 2001.
- [MP04] G. McGraw and B. Potter: *Software Security Testing*, IEEE Security and Privacy, volume 2(5), (2004), pages 81–85.
- [MW06] S. J. Metsker and W. C. Wake: *Design Patterns in Java*, Addison Wesley Professional, 2nd edition, 2006.
- [Off10] Office of Governance Commerce: *IT Infrastructure Library*, <http://www.itil.org>, 2010 (accessed July 8, 2010).
- [Ora10] Oracle Corporation - Sun Developer Network: *Secure Coding Guidelines for the Java Programming Language, Version 3.0*, <http://java.sun.com/security/seccodeguide.html>, 2010.
- [RAF04] N. Rutar, C. B. Almazan and J. S. Foster: *A Comparison of Bug Finding Tools for Java*, Software Reliability Engineering, International Symposium on, volume 0, (2004), pages 245–256.
- [RFMPG06] D. G. Rosado, E. Fernandez-Medina, M. Piattini and C. Gutierrez: *A Study of Security Architectural Patterns, ARES '06: Proceedings of the First International Conference on Availability, Reliability and Security*, IEEE Computer Society, Washington, DC, USA, 2006, pages 358–365.

- [San94] R. S. Sandhu: *On Five Definitions of Data Integrity, Proceedings of the IFIP WG11.3 Working Conference on Database Security VII*, North-Holland Publishing Co., Amsterdam, The Netherlands, The Netherlands, 1994, pages 257–267.
- [SFBH⁺06] M. Schumacher, E. Fernandez-Buglioni, D. Hybertson, F. Buschmann and P. Sommerlad: *Security Patterns - Integrating Security and Systems Engineering*, Wiley Series in Software Design Patterns, John Wiley & Sons, Ltd, 2006.
- [VM04] D. Verdon and G. McGraw: *Risk Analysis in Software Design*, IEEE Security and Privacy, volume 2(4), (2004), pages 79–84.
- [WE10] G. Weirich and Elexis: *Elexis – Die Elektronische Praxis*, <http://www.elexis.ch>, 2010 (accessed July 7, 2010).
- [WH06] R. Wolter and K. Haselden: *Building Distributed Applications - The What, Why, and How of Master Data Management*, msdn.microsoft.com, 2006.
- [Wha10] WhatIs.com: *What is Component? - Definition from WhatIs.com*, <http://whatistechtarget.com>, 2010 (accessed July 7, 2010).
- [Whe03] D. Wheeler: *Secure Programming for Linux and Unix HOWTO – Creating Secure Software*, <http://www.dwheeler.com/secure-programs>, 2003.
- [ZG00] M. Zviran and C. Glezer: *Towards generating a data integrity standard*, Data & Knowledge Engineering, volume 32(3), (2000), pages 291–313.