

# Chapter 2

## Background

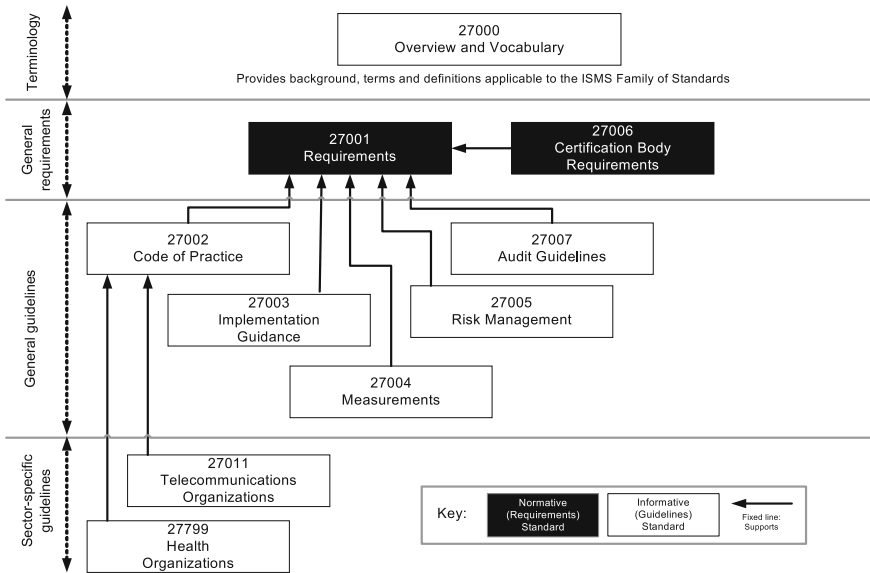
**Abstract** The background that is required to follow the remainder of this book is provided in this chapter. We provide descriptions of security standards, which are supported by various methods and approaches presented in this book, namely the ISO 27001 and the Common Criteria. In addition, we show how to apply our research to the safety, as well. In contrast to security standards that are concerned with protecting a system from attackers, safety standards aim to prevent harm to humans arising from hazards. The safety standard ISO 26262 focuses on the automotive domain and we introduce the standard in this chapter, as well. This book provides methods and techniques of how to apply requirements engineering methods to the establishment of security and safety standards. Hence, we introduce a conceptual framework for security requirements engineering and several of these methods in detail such as Si\* and CORAS. Finally, we show the agenda approach, which is the underlying conceptual foundation of all methods contributed by this book.

### 2.1 Overview

We describe the required knowledge about security standards (Sect. 2.2), the ISO 26262 safety (Sect. 2.3) standard, a framework for security requirements engineering methods (Sect. 2.4), several security requirements engineering methods (Sect. 2.5), and the Agenda Concept (Sect. 2.6) in this chapter.

### 2.2 Security Standards

In this section we introduce the security standards that are the basis for our work, namely the ISO 27001 (Sect. 2.2.2) and the Common Criteria (Sect. 2.2.4). The ISO 27001 is the mandatory standard of the ISO 27000 series of standards (Sect. 2.2.1). The fundamental difference between these standards is that ISO 27001 describes a process for security management, while the Common Criteria describes how to document a software product.



**Fig. 2.1** ISO 27000 overview taken from (ISO/IEC 2009) (see footnote 1)

### 2.2.1 The ISO 27000 Series of Standards

The ISO 27000 series of standards addresses information security matters (ISO/IEC 2009) and the resulting Information Security Management System (ISMS). This is a system independent of vendors, technologies, or the size/type of organization that is part of the management system of an organization (Calder 2009).

The ISO 27000 series of standards is still in development. This means that numerous standards are not published yet. The standards in this family can be divided into several categories. The standards 27000–27005 concern the general description of an ISMS. The mandatory standard in the series is the ISO 27001 (ISO/IEC 2005) that defines how to build an ISMS. Certification of an implementation of the ISO 27001 process is possible. All the other standards of the series are specializations of this standard and describe parts or usage scenarios of the ISMS in detail (ISO/IEC 2009).

The ISO 27000 standard (ISO/IEC 2009) divides the standards of the ISO 27000 series of standards into four categories (see Fig. 2.1<sup>1</sup>).

The ISO 27000 standard itself defines the terminology of the series, the ISO 27001 states the general requirements for an ISMS. General guidelines specify parts of the ISMS, e.g., the ISO 27005 specifies risk management. Sector-specific guidelines

<sup>1</sup>Reproduced by permission of DIN Deutsches Institut für Normung e.V. The definitive version for the implementation of this standard is the edition bearing the most recent date of issue, obtainable from Beuth Verlag GmbH, Burggrafenstraße 6, 10787 Berlin, Germany.

describe how an ISMS is to be implemented in a specific kind of organization, e.g., ISO 27011 concerns telecommunication organizations.

ISO 27000 provides a general overview and the vocabulary used in the ISO 27000 family. The following standards are also part of the ISO 27000 series of standards. ISO 27002 is a so-called *code of practice* and describes controls that can be used in an ISMS implementation.<sup>2</sup> ISO 27003 provides guidance in respect to project management when introducing an ISMS in an organization. ISO 27004 describes measurements for the effectiveness of an ISMS and ISO 27005 concerns risk management. In addition, the risk management in ISO 27005 uses the ISO Guide 73 that defines basics of risk management and the ISO 31000 family that exclusively addresses risk. ISO 31000 describes general principles and measurements and ISO 31010 presents risk assessment techniques (Klipper 2010). The ISO 27007 describes the auditing and certification of an ISMS developed in compliance with the ISO 27001 standard, while the ISO 27006 lists the certification body requirements. Organizations can get accreditation for certifying ISO 27001 realizations.

The remaining standards of the series describe a specific topic in relation to the ISMS. For instance, ISO 27010 describes how to combine different ISMS within one company, ISO 27031 describes business continuity management. The following standards are not published yet. ISO 27032 will provide information for an ISMS with specific types of software applications, e.g., Web 2.0, Software as a Service, and Blogging. ISO 27033 will provide information for the network security part of an ISMS, while ISO 27034's aim is application security. ISO 27035 will describe the incident management and ISO 27036 will provide guidelines for outsourcing (Klipper 2010).

However, even though numerous standards in the ISO 27000 series exist that specialize parts of the ISO 27001, it is not mandatory to use these specializations. The standard also allows to use different specifications, as long as they fulfill the requirements of the ISO 27001 (Klipper 2010). Hence, we focus in our work on the ISO 27001 standard.

The ISO considers also to publish a standard ISO 27017 to provide guidance for implementing an ISMS for clouds and the ISO 27018 to provide privacy guidelines for clouds. Both standards will be released in a draft status soon. However, neither of them will replace the ISO 27001 as the normative standard of the ISO 27000 series of standards.<sup>3</sup> Organizations can get accreditation for certifying an ISO 27001 implementation. However, the German Bundesamt für Sicherheit in der Informationstechnik (BSI) also provides a certification of ISO 27001 based upon their Grundschutz approach (BSI 2011).<sup>4</sup> In practical terms this means that the certification in this case is based on the BSI Grundschutz standards (BSI 2011). Nevertheless, the BSI certification is not accredited by the ISO (Klipper 2010).

---

<sup>2</sup> The standard was formerly known as ISO 17799 and later renamed to ISO 27002.

<sup>3</sup> Online Statement of the ISO 27000 series that ISO 27001 will remain the only mandatory standard of the series: <http://www.iso27001security.com/html/27017.html>.

<sup>4</sup> Note: This is the Bundesamt für Sicherheit in der Informationstechnik (BSI), a national government body that aims to increase IT security. This is not The British Standards Institution (BSI).

### 2.2.2 ISO 27001

The ISO 27001 defines the requirements for establishing and maintaining an ISMS (ISO/IEC 2005). In particular, the standard describes the process of creating a model of the entire business risks of a given organization and specific requirements for the implementation of security controls.

The ISO 27001 standard is structured according to the “Plan-Do-Check-Act” (PDCA) model, the so-called *ISO 27001 process* (ISO/IEC 2005). In the *Plan* phase an ISMS is established,<sup>5</sup> in the *Do* phase the ISMS is implemented and operated, in the *Check* phase the ISMS is monitored and reviewed, and in the *Act* phase the ISMS is maintained and improved. In the *Plan* phase, the *scope and boundaries* of the ISMS, its *interested parties, environment, assets*, and all the *technology* involved are defined. In this phase also the *ISMS policies, risk assessments, evaluations, and controls* are defined. Controls in the ISO 27001 are measures to *modify risk*. The ISO 27005 (ISO/IEC 2008) refines this process for risk management and extends it with a pre-phase for information gathering. The ISO 27001 standard demands a set of documents that describe how the requirements for the ISMS are fulfilled by a concrete implementation.

Changes in the organization or technology also have to comply with the documented ISMS. Furthermore, the standard demands periodic audits towards the effectiveness of an ISMS. These audits are also conducted using documented ISMS requirements. In addition, the ISO 27001 standard demands that management decisions, providing support for establishing and maintaining an ISMS, are documented as well.

The standard demands a set of documents for certification. In the following we list these documents, giving them names in order to simplify the reference to them later in the chapter.

1. The *Scope of the ISMS*;
2. the *ISMS Policy Statements* that contain general directions towards security and risk;
3. the *Procedures and Controls in Support of the ISMS*;
4. a description of the applied *Risk Assessment Methodology*;
5. a *Risk Assessment Report*;
6. a *Risk Treatment Plan*;
7. documented *Procedures to the effective planning, operation and control of the ISMS*;
8. *ISMS Records* that can provide evidence of compliance to the requirements of the ISMS;
9. the *Statement of Applicability* describing the control objectives and controls that are relevant and applicable to the organization’s ISMS;
10. the *Management Decisions* that provide support for establishing and maintaining an ISMS.

---

<sup>5</sup> Note that we defined the term establishment with regard to standards in Chap. 1.

Note that ISO 27001 Sections 4.3.2 and 4.3.3 concern the control of documents and records that shall be specified in document (8). We focus on how to create the document and regard the protection of records as future work and do not consider it in this book.

### **2.2.3 ISO 27001:2013**

In the end of 2013 ISO released a new version of the ISO 27001 standard (ISO/IEC 2013), hereafter called ISO 27001:2013. By the end of 2014 all certification bodies will support only the new version of the standard. Moreover, existing certifications of the old version of the standard, which we refer to if we only write ISO 27001, will have to be updated to ISO 27001:2013 within a period of 2–3 years after the release of ISO 27001:2013.<sup>6</sup> This book is being released during the transition phase of these two versions of the standards. Hence, we consider in our work both versions of the standard. We illustrate first how they support the old version and afterwards discuss the required changes to be compliant to the new version. Thus, we illustrate our research to readers that are familiar with the old version and show them how to transition to the new version. Moreover, we illustrate the origins and the evolution of the standard to readers not familiar with the approach. This way they can not only support the new version, but also help transitioning from one version to the next.

The new version of the standard does not explicitly consider the Plan-Do-Check-Act model anymore. Moreover, ISO 27001:2013 is compliant to a structure for management system standards defined by ISO in Annex SL of ISO/IEC Directives (ISO/IEC 2013, cf. p.1). The ISO 27001:2013 is also closely aligned with ISO 31000 (ISO 2009), a standard for risk management. This has led to changes in terminology, which we explain in Chap. 5.

The structure of ISO 27001:2013 begins with a scope definition of the standard, states normative references to other standards, in this case to the new version of ISO 27000 from 2014 (ISO/IEC 2014). Followed by a section for terms and definition, which just contains a reference to the definitions in ISO 27000 (ISO/IEC 2014). The next section demands a context description of the organization aligned with ISO 31000 (ISO 2009). Afterwards the leadership commitments have to be defined. The following section concerns the risk management and security objectives of an ISMS. The remaining sections concern support, operation, evaluation, and improvement of an ISMS. The new version contains also the mandatory Annex A with security controls.

---

<sup>6</sup> <http://www.bsigroup.com/en-GB/iso-27001-information-security/ISOIEC-27001-Revision/>.

ISO 27001:2013 does not define document types that have to be written, but defines so-called *documentation information* in its sections. The following sections contain documentation information:

1. Section 4.3 demands documentation information about the *Scope of the ISMS*
2. Section 5.2 demands documentation information about the *Information Security Policy*
3. Section 6.1 demands documentation information about the *Risk Assessment* and the *Risk Treatment* including the *Statement of Applicability*
4. Section 6.2 demands documentation information about the *Information Security Objectives*
5. Section 7.2 demands documentation information about the *Competence Records*
6. Section 8.2 demands documentation information about the *Risk Assessment Results*
7. Section 8.3 demands documentation information about the *Risk Treatment Results*
8. Section 9.1 demands documentation information about the *Monitoring and Measuring Results*
9. Section 9.2 demands documentation information about the *Audit Programme and Results*
10. Section 9.3 demands documentation information about the *Management Review Results*
11. Section 10.1 demands documentation information about the *Evidence of Corrective Actions*.

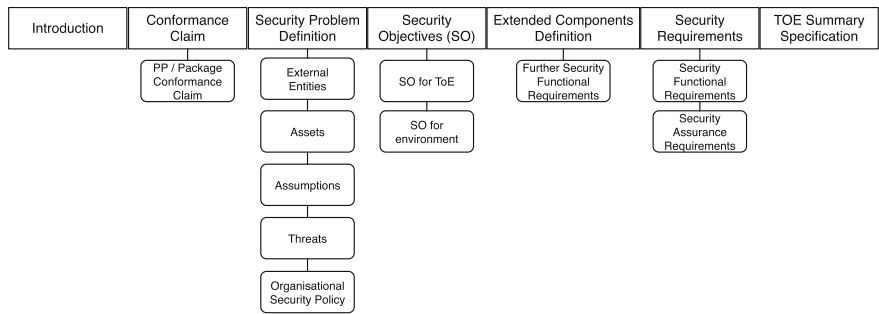
The documentation can contain further documents if some topics have to be elaborated further, as suggested by e.g., Sections 7.5 and 8.1.

### **2.2.4 Common Criteria**

The ISO/IEC 15408—Common Criteria for Information Technology Security Evaluation—(ISO/IEC 2012) is a security standard that can achieve comparability between the results of independent security evaluations from different security analysts of IT products. These are so-called *targets of evaluation (ToEs)*.

The Common Criteria is based on a general security model. The model considers ToE owners that value their assets and wish to minimize risk to these assets via imposing countermeasures. These reduce the risk to assets. Threat agents want to abuse assets and give rise to threats for assets. The threats increase the risk to assets.

The concepts of the Common Criteria consider that potential ToE owners infer their security needs for specific types of ToE, e.g., a specific database. The resulting documents are called Security Targets (ST). Protection profiles (PP) state security needs for an entire class of ToEs, e.g., client VPN application. The evaluators check if a ToE meets its ST. PPs state the security requirements of ToE owners. ToE developers or vendors publish their security claims in an ST. A CC evaluation determines if the



**Fig. 2.2** The structure of a CC security target/protection profile (inspired by Fabian et al. 2010)

ST is compliant to a specific PP. The standard relies upon documents for certification, which state information about security analysis and taken measures.

The structure of a CC security target, depicted in Fig. 2.2 inspired by Fabian et al. (2010), starts with an ST *Introduction* that contains the description of the ToE and its environment. The *Conformance Claims* describe to which PPs the ST is compliant. The *Security Problem Definition* refines the external entities, e.g., stakeholders in the environment and lists all assets, assumptions about the environment and the ToE, threats to assets and organizational security policies. The *Security Objectives* have to be described for the ToE and for the operational environment of the ToE. The *Extended Component Definitions* describe extensions to security components described in the CCs part 2. The *Security Requirements* contain two kinds of requirements. The security functional requirements (SFR) are descriptions of security functions specific to the ToE. The security assurance requirements (SAR) describe the measures taken in development of the ToE. These are evaluated against the security functionality specified in the SFR. The Evaluation Assurance Level (EAL) is a numerical rating ranging from 1 to 7, which states the depth of the evaluation. Each EAL corresponds to an SAR package. EAL 1 is the most basic level and EAL 7 the most stringent.

2.3 Safety Standard ISO 26262

We propose a hazard analysis method based on problem frames (Sect. 2.5.3) compliant to ISO 26262 (Chap. 9). This section introduces this standard. ISO 26262 (ISO 2011) is a risk-based functional safety standard concerning safety-related systems that include one or more Electric and Electronic (E/E) systems, which are installed in passenger cars with a max gross weight up to 3500 kg. It addresses possible hazards caused by malfunctions of E/E safety-related systems, including the interaction of these systems and their subsystems.

ISO 26262 is derived from the generic functional safety standard ISO/IEC 61508 (International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC) ISO/IEC 2000). It is aligned with the automotive safety lifecycle including specification, design, implementation, integration, verification, validation, configuration, production, operation, service, decommissioning, and management. ISO 26262 provides an automotive-specific risk-based approach for determining risk classes, which describe the necessary risk reduction for achieving an acceptable residual risk, the so-called *automotive safety integrity level (ASIL)*. The defined ASILs are *QM*, *ASIL A*, *ASIL B*, *ASIL C*, and *ASIL D*. ASIL D requires the highest risk reduction, while for functions with ASIL A, ASIL B, or ASIL C, fewer requirements on the development processes, safety mechanisms, and evidences are necessary. ISO 26262 demands just the normal quality measures applied in the automotive industry for a QM rating.

## 2.4 A Conceptual Framework for Security Requirements Engineering

Notions and terminology differ in different SRE methods (Fabian et al. 2010). In order to be able to compare different SRE methods, Fabian et al. (2010) developed a Conceptual Framework (CF) that explains and categorizes building blocks of SRE methods. In their survey the authors also use the CF to compare different SRE methods. Karpati et al. (2011) conclude in their survey that the only existing “uniform conceptual framework for translations” of security terms and notions for SRE methods is the work of Fabian et al. (2010). Therefore, we use this CF and base our relations between the ISO 27001 and SRE methods on it. We work on a subset of the CF (see Fig. 2.3 inspired by Fabian et al. 2010) that focuses on concerns relevant for our work. We derived this subset during the research done for this book.

The CF considers security as a system property using the terminology of Jackson (2001) as introduced in Sect. 2.5.3. The CF considers four main building blocks of SRE methods: *Stakeholder Views*, *System Requirements*, *Specification and Domain Knowledge*, and *Threat Analysis*, as depicted in Fig. 2.3. *Stakeholder Views* identify and describe the stakeholders and their functional and nonfunctional goals and resulting functional and nonfunctional requirements. Stakeholders express security concerns via security goals. These goals are described toward an asset of the stakeholder, and they are refined into security requirements. *System Requirements* result from a reconciliation of all functional, security, and other nonfunctional requirements, while the stakeholder view perspective focuses on the requirements of one stakeholder in isolation. Hence, the system requirements analysis includes the elimination of conflicts between requirements and their prioritization. The result is a coherent set of system requirements. Requirements are properties the system has after the machine is built. The *Specification and Domain Knowledge* building block consists of specifications, assumptions, and facts. The specification is the description of the interaction behavior of the machine with its environment. It is the basis for the



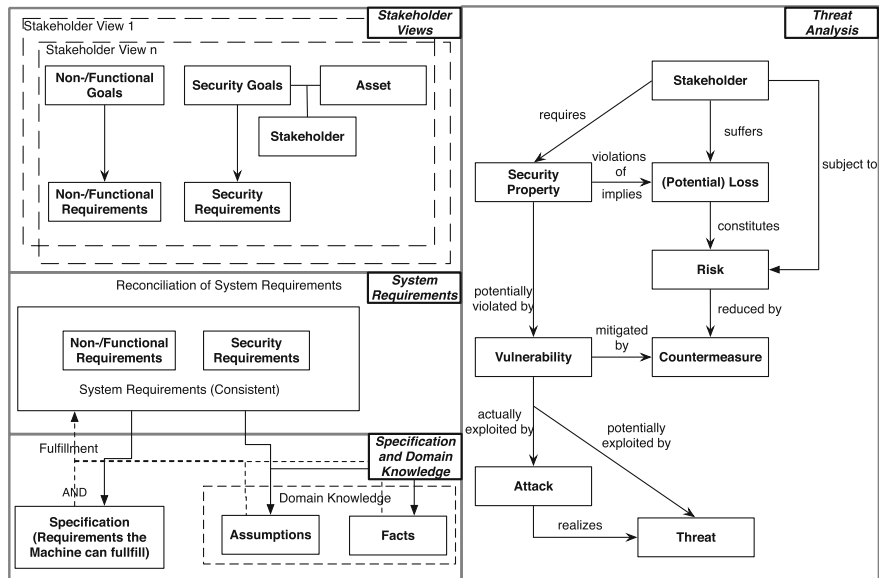


Fig. 2.3 The conceptual framework for security requirements engineering (inspired by Fabian et al. 2010)

construction of the machine. Assumptions and facts make up the domain knowledge. The domain knowledge describes the environment in which the machine will be integrated. In practical terms this means the security requirements have to be reviewed in context of the environment. The *Threat Analysis* focuses on security properties required by stakeholders. A violation of a security property is a potential loss for a stakeholder. This loss constitutes a risk for the stakeholder, which is reduced by countermeasures. A vulnerability may lead to a violation of a security property, and it is mitigated by a countermeasure. Attacks actually exploit vulnerabilities, while threats only potentially exploit vulnerabilities. Attacks realize threats.

**Relations Between the Common Criteria and the Conceptual Framework for Security Requirements Engineering**

The relations of the Common Criteria with the CF were analyzed by Fabian et al. (2010). We show the relation to the CF in Table 2.1. Fabian et al. (2010) described the relations in text and we show the relations in a table. We listed important terms of the conceptual framework in the left column of this table and stated the equivalent term(s) on the right. If both terms are identical, we listed a “~” in the table. The Common Criteria views security requirements as functional (or assurance) requirements. Other functional or nonfunctional requirements are not considered if they are not relevant to the security functionality, but the standard does not define a clear cut decision criteria for a relevant security functionality. In addition, requirements conflicts are not explicitly considered.

**Table 2.1** Correspondence: terms of common criteria and the CF (inspired by Fabian et al. 2010)

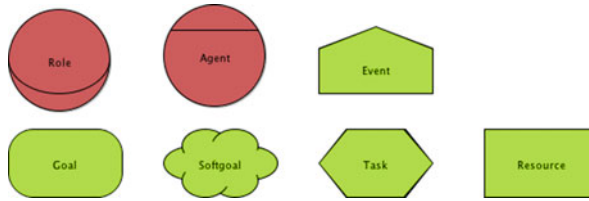
CF Fabian et al.	Common criteria
System	~
Machine	The target of evaluation (ToE) is the software/hardware, which should be evaluated
Environment	~
Security goal	The common criteria considers high-level <i>security needs</i> of ToE owners
Security requirement	The <i>security objective</i> in a protection profile is a refined security need, which is similar to a security requirement being a refined security goal in the terminology of the CF
Specification	The <i>security objective</i> in the security target document describes security solutions, which are selected to address the threats. <i>Security Functional Requirements (SFRs)</i> are textual patterns in the common criteria, which have to be instantiated for a particular ToE. These refine the solutions of the elicited security objectives for the ToE
Stakeholder	The ToE owner of the common criteria's general security model and users in SFRs are stakeholders in the common criteria
Domain knowledge	The common criteria considers <i>assumptions</i> about the environment and also specific <i>security objectives</i> for the operational environment of the ToE, e.g., a securely configured firewall
Availability	~
Confidentiality	~
Integrity	~
Asset	~
Threat	~
Vulnerability	~
Risk	~

## 2.5 Security Requirements Engineering Methods

We introduce requirements engineering methods, which we enhance in our research (Chaps. 5–8). Note that the problem frame-based methods in Sect. 2.5.3 have security specific extensions, but are also useful for software engineering in general.

### 2.5.1 *Si*\*

The *Si*\* modeling language (Massacci et al. 2010; Asnar et al. 2011) has been proposed to capture security and functional requirements of socio-technical systems. *Si*\* is founded on the concepts of *agent*, *role*, *goal*, *task*, *resource*, depicted in Fig. 2.4.

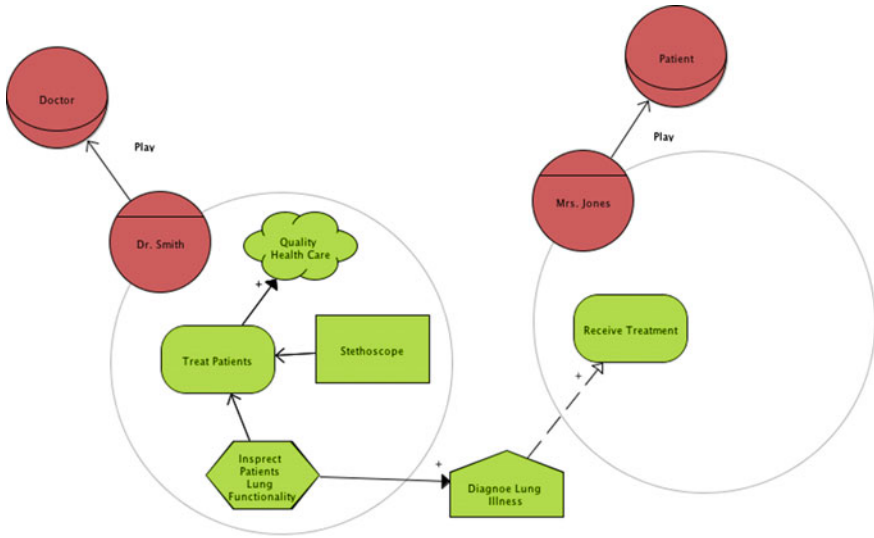


**Fig. 2.4** Elements of the Si\* notation

An agent is an active entity with concrete manifestations and is used to model humans as well as software agents and organizations. A role is the abstract characterization of the behavior of an active entity within some context. They are graphically represented as circles. Assignments of agents to roles are described by the *play* relation. The term actor refers to role and agent alike and is used in cases where these are not distinguished. A goal is a state of affairs whose realization is desired by some actor (objective), can be realized by some (possibly different) actor (capability), or should be authorized by some (possibly different) actor (entitlement). Soft goals are similar but have no clear criteria for stating if they are fulfilled or not. A task specifies the procedure used to achieve goals. A resource represents a physical or an informational entity without intentionality. A resource can be consumed or produced by a task. Events represent uncertain circumstances that have an impact on the fulfillment of goals or cause security concerns of a resource. In the graphical representation, goals, tasks, resources, and events are, respectively, represented as ovals, hexagons, rectangles, and pentagons. *Contribution* relations are used when the relation between goals is not the direct consequence of a deliberative planning but rather results from side effects. These relations have black arrowheads and a solid line. The impact can be positive or negative and is graphically represented as edges labeled with + and –, respectively. *Impact* relations describe the effect a task or an event has on a resource or a goal. These relations have white arrow heads and a dashed line. Negative impacts are denoted with the sign “–”, and for significant negative impact with the sign “—”. Positive impacts are opportunities and denoted with the sign “+”, and for significant positive impact with the sign “++”. Finally, tasks or resources are linked to the goals that they intend to achieve using *means-end* relations. The relations have a solid line and a dart shaped arrowhead.

We provide an example of the Si\* notation in Fig. 2.5. The example concerns the medical domain. *Dr. Smith* plays the role *Doctor*. *Dr. Smith* contributes to the soft goal *Quality Health Care* by his goal to *Treat Patients*. The resource *Stethoscope* is a means-end to this goal, as well as the task to *Inspect Patients Lung Functionality*. This task has a positive contribution on the event *Diagnose Lung Illness*. The event represents an uncertain circumstance that has a positive impact on the *Receive Treatment* goal of *Mrs. Jones*, who plays a *Patient*.

Goals and tasks of the same actor or of different actors are often related to one another in many ways. AND/OR decomposition combines AND and OR refinements



**Fig. 2.5** Example healthcare in Si\*

of a root goal into subgoals. However, neither are such goals possibly under the control of the actor, nor does the actor may have the capabilities to achieve them.

We illustrate a goal decomposition in Fig. 2.6. The role *Pharmacy* has the goal to *Sell Drugs*, which is decomposed into the goals *Provide Drugs* and *Manage Prescriptions*.

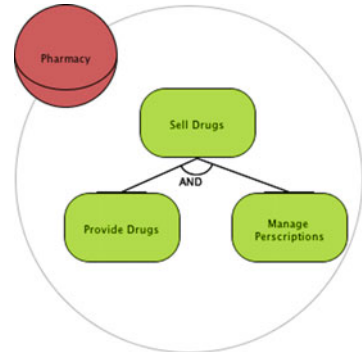
The relations between actors within the system are captured by the notions of *delegation* and *trust*. Assignment of responsibilities among actors can be expressed by *execution dependency*, when one actor depends on another actor for the achievement of a goal. Assignment of responsibilities among actors can also be expressed by *permission delegation*, when an actor authorizes another actor to achieve the goal. Usually, an actor prefers to appoint actors that are expected to achieve assigned duties and not misuse granted permissions. In the graphical representation, permission delegations are represented with edges labeled by **Dp** and execution dependencies with edges labeled by **De**.

Entitlements and capabilities of actors with regard to resources are modeled using the so-called *ECO Model* in Si\*. The *own* relationship indicates that an actor has the property rights over a resource; *provide* indicates that an actor has the capabilities to make this resource available to another role or actor.<sup>7</sup> Own and provide are represented with edges between an actor and a resource labeled by **O** and **P**, respectively.

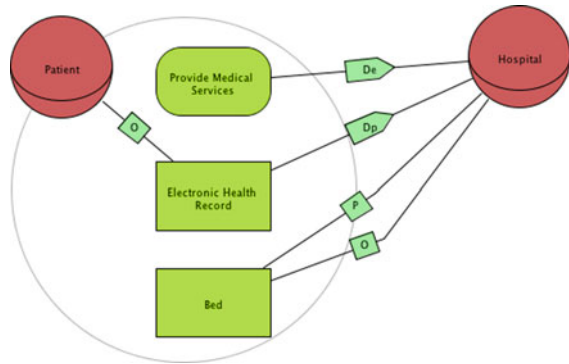
Figure 2.7 shows the application trust and ECO Model relations. A *Patient* has the goal to *Provide Medical Services* and the *Patient* applies a delegation execution

<sup>7</sup> Note that in contrast to our work Massacci applies the ECO Model also to relations between actors and goals.

**Fig. 2.6** Example Si\* goal decomposition



**Fig. 2.7** Example trust relations and ECO model relations in Si\*

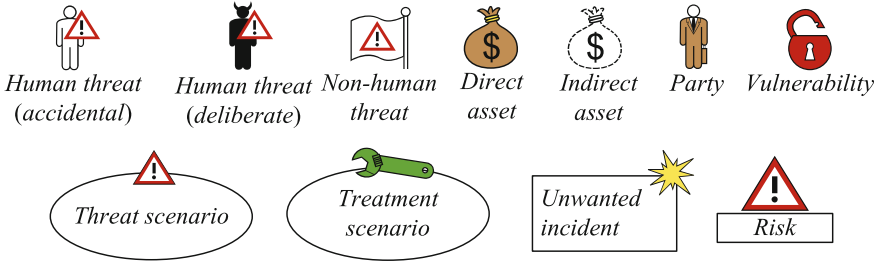


to the *Hospital* of this goal. The *Patient* owns an *Electronic Health Record* that contains his/her medical information. The *Patient* gives a permission delegation to the *Hospital* to access and use the *Electronic Health Record*. In addition, the *Hospital* owns a *Bed*, which is provided to the *Patient* by the *Hospital*.

## 2.5.2 CORAS

The CORAS method (Lund et al. 2010) is a model-driven approach to risk analysis and comes with a method, a language to support all steps of the method, as well as a tool that is used throughout the process to conduct the tasks and document the results. CORAS is classified as a risk-based security requirements engineering method, because it works on a similar abstraction level (see the survey from Fabian et al. 2010).

The method follows the five steps of ISO 31000 (ISO 2009), which are *context establishment*, *risk identification*, *risk estimation*, *risk evaluation* and *risk treatment*. The three activities in the middle are referred to as *risk assessment*.



**Fig. 2.8** CORAS symbols taken from Lund et al. (2010)

The symbols used in CORAS diagrams are depicted in Fig. 2.8. The elements asset, vulnerability, and threat are essential parts of any risk analysis. Assets are the things of value to a party that a threat can harm by exploiting a vulnerability. A threat scenario states how a threat exploits a certain vulnerability and leads to an unwanted incident. This incident causes harm to an asset. The likelihood or consequence of an unwanted incident result in a risk. If a risk is not acceptable, a treatment scenario has to be implemented that reduces the likelihood or consequence of the unwanted incident that causes the risk. Threats can arise from humans in an accidental or deliberate way. Threats can arise from a nonhuman entity as well. Note that we focus on security analysis in this work and on deliberate human threats.

### Step 1: Context Establishment

Context establishment includes defining the scope and focus of the analysis, modeling the target of analysis at an adequate level of abstraction, identifying stakeholders and assets, and defining the risk evaluation criteria. The target is modeled using a (semi-) formal language, such as UML (UML Revision Task Force 2010), and assets are documented using CORAS *asset diagrams*. Figure 2.9 illustrates an example target description. This example concerns an eHealth scenario. A *Patient* of a hospital stores his medical information, e.g., X-rays, previous illnesses, current medication, and further information in an Electronic Health Record (EHR). The EHR is accessed using the *Hospital Information System* (HIS). The HIS and EHR are used by a *Nurse* and a *Doctor*. In addition, the HIS communicates with a *Local Physician*, who is out of scope of this analysis.

We show an example of an asset diagram in Fig. 2.10. The customer that conducts the risk analysis is the *Hospital* that evaluates the use of electronic health records. The risk analysis concerns two assets. The *Electronic Health Records* and the *Public's Trust in Electronic Health Records*. The arrow between the assets represents a *harm to* relationship, meaning harm to *Electronic Health Records* causes also harm to the *Public's Trust in Electronic Health Records*. CORAS considers two types of assets. The harm to a direct asset can be measured, while the harm to an indirect asset cannot. In addition, we can implement treatments for threats for direct assets. The reason for the harm to relation is to identify the direct assets that cause harm to the indirect ones. Treatments to the direct asset shall support risk reduction for indirect

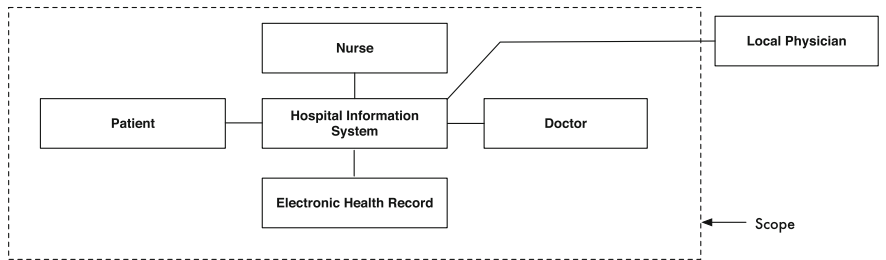


Fig. 2.9 The semi-formal target description of an eHealth scenario

Table 2.2 Example high-level risk table

Who or what causes it?	How? What is the incident? What does it harm?	What makes it possible?
Hacker	System break-in and theft of electronic health record	Insufficient security

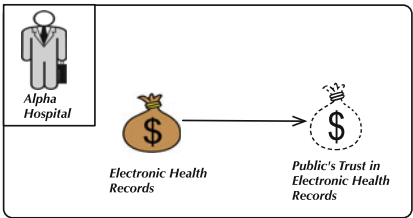
assets, as well. For example, if the confidentiality of the Electronic Health Records is breached and numerous incidents have been reported, the asset is harmed. This would lead to a reduction in the Public’s Trust in Electronic Health Records.

The context establishment also involves a high-level risk identification as part of defining the scope and focus. This is done using so-called *High-level Risk Tables*. We show an example in Table 2.2. The table states the cause of a threat in the first column, the incident and harm in the second column, and the issue that makes the incident possible in the third. The information for the table is the result of expert interviews and other sources of knowledge.

Step 2: Risk Identification

Risk identification is conducted by the identification, modeling, and documentation of threats, threat scenarios, vulnerabilities, and unwanted incidents with respect to the target of analysis and the identified assets. The modeling is conducted using CORAS *threat diagrams*. Figure 2.11 illustrates an example of a threat diagram. In this diagram a *Hacker* exploits the vulnerability *Insufficient Security* and causes the threat scenario *System Break-in and eavesdropping on Electronic Health Records*, which

Fig. 2.10 CORAS asset diagram



**Table 2.3** Example qualitative likelihood scale

Likelihood value	Description
Likely	A significant number of similar incidents have been recorded
Possible	Several similar incidents on record

**Table 2.4** Example qualitative consequence scale

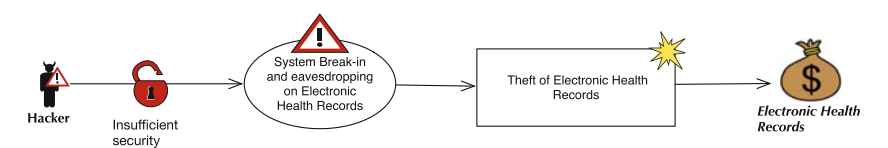
Consequence	Generic interpretation
Major	Large-scale thefts have happened; can cause significant monetary reparations for the hospital
Moderate	Several thefts on a small scale have happened; can cause monetary reparations for the hospital

leads to the unwanted incident *Theft of Electronic Health Records*. This incident harms the asset *Electronic Health Record*.

Risk assessment can be conducted either quantitative or qualitative. Quantitative risk assessment demands that the likelihood and consequences scales contain numeric values. These have to express in which time frame a risk is likely and what the consequences are in, e.g., number of affected assets. Should these numbers not be available, because the system has not been built yet, likelihood and consequences tables can contain a qualitative scale that does not contain numbers. This is a starting point for risk assessment, and should the numeric values become available, a quantitative risk assessment should be done. We present likelihood (see Table 2.3) and consequence scales (see Table 2.4) for our example.

Step 3: Estimate Risk

The risk estimation is also conducted using threat diagrams, and involves the estimation of likelihoods and consequences for the identified incidents and scenarios. Figure 2.12 illustrates an annotated threat diagram with likelihoods and consequence. The likelihood of the threat scenario is *likely* and the likelihood of the unwanted incident is *possible*. These values are picked at random for this example. The consequence of the unwanted incident is *major* for the Electronic Health Records, because these



**Fig. 2.11** CORAS threat diagram



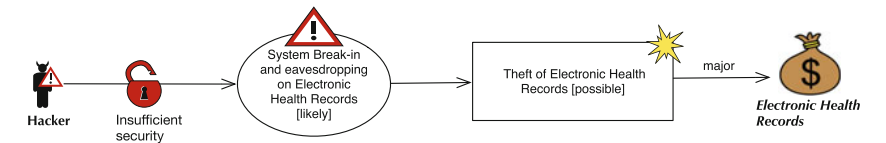
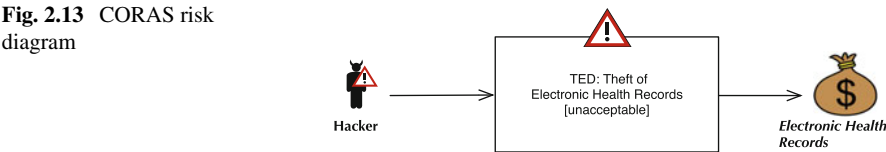


Fig. 2.12 CORAS threat diagram with likelihoods and consequence

Fig. 2.13 CORAS risk diagram



contain the illnesses of a person that shall stay confidential in order to avoid possible blackmail attempts.

We present a risk evaluation matrix for our example (see Table 2.5). The white fields in the matrix represent fields that are acceptable combinations of likelihood and consequences. Gray fields on the other hand are unacceptable risks that have to be treated.

Step 4: Evaluate Risk

Risk evaluation is also conducted using CORAS risk diagrams, and includes determining which risks need to be considered for possible treatment by comparing the risks against the evaluation criteria. Figure 2.13 shows a risk diagram for our example. The digram shows that a Hacker causes the risk TED: Theft of Electronic Health Records. The risk is classified as unacceptable and concerns the asset Electronic Health Records.

Step 5: Risk Treatment

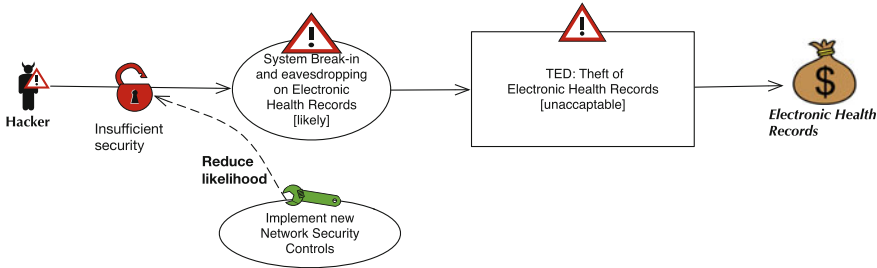
Finally, risk treatment is conducted to identify means to mitigate unacceptable risks, and is conducted using CORAS treatment diagrams. Our example in Fig. 2.14 shows a treatment diagram. The treatment is to Implement new Network Security Controls and it shall reduce the likelihood of a Hacker exploiting the vulnerability Insufficient security. Treatments can also be shown in treatment overview diagrams, which show only the threat, risk, asset, and treatments. They do not show vulnerabilities or threat scenarios.

Legal CORAS

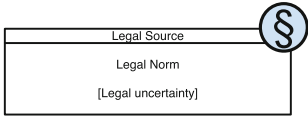
Legal CORAS (Lund et al. 2010) is an extension of CORAS specifically for considering legal aspects and legal risk. The approach is based on existing work on legal risk

Table 2.5 Example risk evaluation matrix

		Consequence	
		Minor	Major
	Likelihood	Likely	TED
	Possible		



**Fig. 2.14** CORAS treatment diagram



**Fig. 2.15** The legal aspects symbol in CORAS

management (Mahler 2010). The initial target description in Legal CORAS contains a statement about whether and to what extent legal aspects should be considered in the risk analysis. The method elicits relevant legal aspects based upon the final target description.

The source of legal risks are legal norms, which are norms that stem from legal sources such as laws, regulations, contracts, and legally binding agreements. When assessing legal risks, there are two kinds of uncertainties that must be estimated. First, the legal uncertainty is the uncertainty of whether a specific norm actually applies to circumstances that may arise. Second, the factual uncertainty is the uncertainty of whether these circumstances will actually occur, and thereby potentially trigger the legal norm. It is by combining the estimates for these two notions of uncertainty that we can estimate the significance of a legal norm and its impact on the risk picture. Legal CORAS comes with the necessary analysis techniques and modeling support, but the involvement of a lawyer or other legal experts is usually required.

Legal CORAS introduces a new symbol to represent legal aspects, depicted in Fig. 2.15. The symbol states first the legal source, e.g., a law. Second, the legal norm, e.g., a part of the law and finally the legal consequence.

We show how legal aspects are integrated into a threat diagram in Fig. 2.16. The legal aspects are modeled as an additional information before the unwanted incident, which is a result of the legal aspects. In our example, we name the German Federal Data Protection Act and the incident is that the Hospital is fined, because a Hacker stole Electronic Health Records.

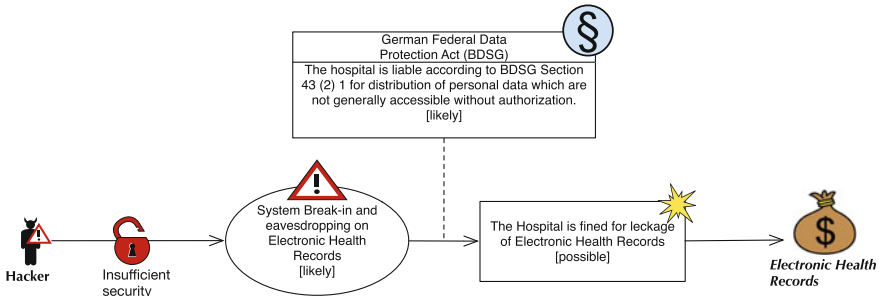


Fig. 2.16 CORAS threat diagram with legal aspects

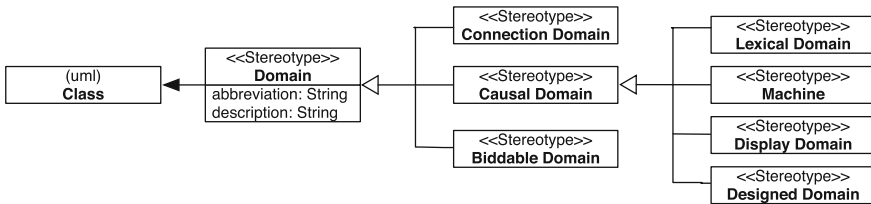


Fig. 2.17 Different domain types (inspired by Côté 2012)

### 2.5.3 Problem Frame-Based Methods

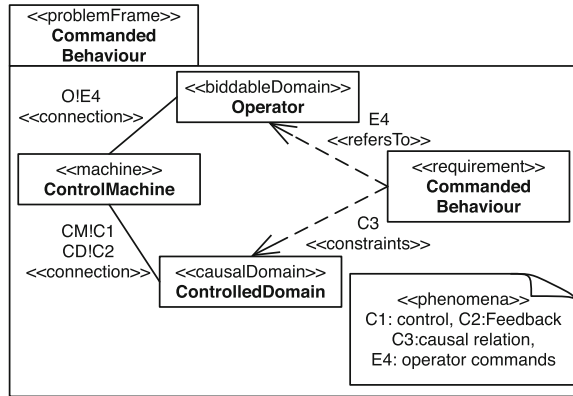
This section introduces a UML-based version of the problem frame notation, which we are using in this book. This section shows the analysis phase of the ADIT development process, which is based on problem frames, as well. We introduce tool support for ADIT in this section.

#### Problem Frames

Problem frames are a means to describe software development problems. They were proposed by Jackson (2001), who describes them as follows: “A problem frame is a kind of pattern. It defines an intuitively identifiable problem class in terms of its context and the characteristics of its domains, interfaces and requirement.” It is described by a *frame diagram*, which consists of domains, interfaces between them, and a requirement. We describe problem frames using class diagrams extended by stereotypes as proposed by Hatebur and Heisel (2010) (see Fig. 2.17). All elements of a problem frame diagram act as placeholders, which must be instantiated to represent concrete problems. In doing so, one obtains a problem description that belongs to a specific kind of problem. Problem frames are an appropriate means to analyze not only functional requirements, but also dependability and other quality requirements (Hatebur and Heisel 2009; Alebrahim et al. 2011).

The UML profile for Jackson’s problem frame notation is called UML4PF. The class with the stereotype *machine* represents the thing to be developed (e.g., the software). The other classes with some domain stereotypes, e.g., *CausalDomain* or *BiddableDomain* represent *problem domains* that already exist in the application

**Fig. 2.18** An example problem frame (taken from Côté 2012)



environment. Domains are connected by interfaces consisting of shared phenomena. Shared phenomena may be events, operation calls, messages, and the like. They are observable by at least two domains, but controlled by only one domain, as indicated by an exclamation mark. For example, in Fig. 2.18 the notation *O!E4* means that the phenomena in the set *E4* are controlled by the domain *Operator*. These interfaces are represented as associations, and the name of the associations contain the phenomena and the domains controlling the phenomena.

Jackson distinguishes the domain types *CausalDomains* that comply with some physical laws, *LexicalDomains* that are data representations, and *BiddableDomains* that are usually people. According to Jackson, domains are either *designed*, *given*, or *machine* domains. The domain types are modeled by the subclasses *BiddableDomain*, *CausalDomain*, and *LexicalDomain* of the class *Domain*. A lexical domain is a special case of a causal domain. This kind of modeling allows one to add further domain types, such as *DisplayDomains* as introduced in Côté et al. (2008) (see Fig. 2.17).

Problem frames support developers in analyzing problems to be solved. They show what domains have to be considered, and what knowledge must be described and reasoned about when analyzing the problem in-depth.

Software development with problem frames proceeds as follows: first, the environment in which the machine will operate is represented by a *context diagram*. Like a frame diagram, a context diagram consists of domains and interfaces. However, a context diagram contains no requirements. Then, the problem is decomposed into subproblems. If ever possible, the decomposition is done in such a way that the subproblems fit to given problem frames. To fit a subproblem to a problem frame, one must instantiate its frame diagram, i.e., provide instances for its domains, phenomena, and interfaces. The instantiated frame diagram is called a *problem diagram*.

Since the requirements refer to the *environment* in which the machine must operate, the next step consists in deriving a *specification* for the machine (see Jackson and Zave 1995 for details). The specification describes the machine and is the starting point for its construction.

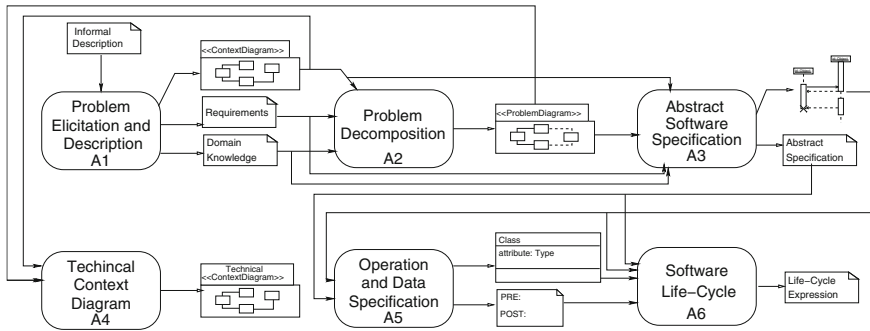


Fig. 2.19 ADIT analysis phase overview (taken from Côté 2012)

### The Analysis Phase of the ADIT Software Development Process

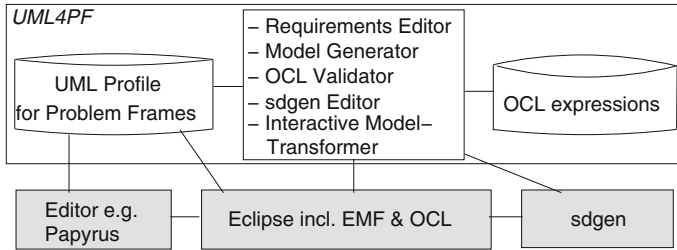
In the following, we describe the development process called ADIT (Analysis, Design, Implementation, and Test). ADIT is a model-driven, pattern-based development process also making use of components. The process is almost based entirely on the UML notation (UML Revision Task Force 2010) and defines relations between different kinds of models. The ADIT process defines consistency checks of these models including tool support (Côté et al. 2011). The ADIT process also supports the traceability between development phases. For example, the process can answer the question which requirements refer to what design artifacts. The process is described in detail in Hatebur (2012) and Côté (2012). We focus on the analysis phase in this work and present this phase in the following. An overview of the different steps for the analysis phase is given in Fig. 2.19 taken from Côté (2012).

**A1—Problem Elicitation and Description** The process begins with a description of the desired functionality of the software to be built. This description is refined into requirements and domain knowledge, which consists of facts and assumptions. We use a *context diagram* using our UML profile and tool support (Côté et al. 2011) that is based on Jackson (2001).

**A2—Problem Decomposition** The second step decomposes the *context diagram* into *problem diagrams* (also according to Jackson 2001). These focus each on one or more requirements and the problem that the requirement expresses.

**A3—Abstract Software Specification** Problem diagrams do not state the order in which the actions, events, or operations occur. They also concern requirements, which refer to problem domains, but not to the machine. Therefore, it is necessary to transform requirements into specifications (see Jackson and Zave 1995 for more details). We use UML *sequence diagrams* (UML Revision Task Force 2010) as models for our specifications. Sequence diagrams describe the interaction of the machine with its environment. Messages from the environment to the machine correspond to *operations* that have to be implemented. The resulting operations will be specified in detail in Step A5.

**A4—Technical Context Diagram** This step describes the technical environment of the machine. For example, a web application machine may use the Apache web server. We use again the UML-based notation mentioned in Steps A1 and



**Fig. 2.20** UML4PF tool realization overview (taken from Côté 2012)

A2. We use a specific kind of context diagram, the technical context diagram. The technical context diagram describe technical means, e.g., on the API of the Apache web server.

**A5—Operation and Data Specification** The purpose of this step is to set up the necessary internal data structures represented as analysis class diagrams. Furthermore, we specify the operations identified in Step Abstract Software Specification by providing pre- and postconditions for each relevant operation. We use OCL (UML Revision Task Force 2010) to express these operation specifications.

**A6—Software Life-Cycle** We use life-cycle expressions proposed by Coleman et al. (1994) to describe the overall behavior of the machine. We express in particular the relations between the sequence diagrams and problem diagrams. This step does not rely on UML, but the notation proposed by Coleman et al. (1994).

### UML4PF Tool Support

UML4PF (Côté et al. 2011) is the UML profile and support tool of the problem frame notation, which is used in the ADIT process (Sect. 2.5.3). The UML<sup>8</sup> profile contains formal validation conditions expressed in OCL<sup>9</sup> and an Eclipse<sup>10</sup> plugin.

Figure 2.20 shows UML4PF's components. White boxes denote components particularly created for UML4PF and gray boxes denote reused components. In the following, we list the UML4PF tool functionalities and items:

- The *UML Profile for Problem Frames* defines the relevant stereotypes for the ADIT method, e.g., `<<ProblemDiagram>>`.
- The *Requirements Editor* provides the functionality to add new requirements.
- The *Model Generator* automatically generates model elements, e.g., observed and controlled interfaces from association names.

<sup>8</sup> <http://www.uml.org/>.

<sup>9</sup> <http://www.omg.org/spec/OCL/2.0/>.

<sup>10</sup> <http://www.eclipse.org/>.

- The *OCL Validator* checks if a model is valid and consistent by evaluating UML4PF's *OCL expressions*. It also returns the names of invalid parts of the model. All in all, UML4PF contains more than 70 OCL validation conditions for the analysis phase.
- The *sdgen Editor* provides the means to edit sequence diagrams.
- The *Interactive ModelTransformer* serves to create software architectures via interactive model transformations.

## 2.6 The Agenda Concept

The *Agenda Concept* (Heisel 1998) concerns process knowledge in the context of software engineering. An agenda is a sequence of steps within a process. These steps produce one or more artifacts, e.g., textual documentation, models, formal expressions, or software. Each step provides methodological support via well-defined *activities*, *inputs*, *outputs*, and *validation conditions*. **Inputs** are descriptions of the essential “ingredients” to perform the activities of a step. **Activities** are descriptions of the processing of all inputs into outputs. A step can contain multiple activities. **Outputs** are the desired results of the activities of this step. **Validation conditions** support the detection of mistakes in outputs. These conditions check the results for, e.g., inconsistencies or mistakes made during the activities. A step can only be completed after the application of all validation conditions with positive results.

Agendas aim at making software engineering knowledge explicit and help engineers to learn and repeat efficient and effective methods for software engineering.

## References

- Alebrahim, A., Hatebur, D., & Heisel, M. (2011). A method to derive software architectures from quality requirements. In *Proceedings of the 18th Asia-Pacific Software Engineering Conference (APSEC)* (pp. 322–330). IEEE Computer Society.
- Asnar, Y., Giorgini, P., & Mylopoulos, J. (2011). Goal-driven risk assessment in requirements engineering. *Requirements Engineering*, 16(2), 101–116.
- BSI. (2011). BSI Grundschrift Homepage. Bonn, Germany: Federal Office for Information Security (BSI). ([https://www.bsi.bund.de/DE/Themen/ITGrundschutz/itgrundschutz\\_node.html](https://www.bsi.bund.de/DE/Themen/ITGrundschutz/itgrundschutz_node.html)).
- Calder, A. (2009). *Implementing information security based on iso 27001/iso 27002: A management guide*. Zaltbommel: Van Haren Publishing.
- Coleman, D., Arnold, P., Bodoff, S., Dollin, C., Gilchrist, H., Hayes, F., et al. (1994). *Object-oriented development: The fusion method*. Englewood Cliffs: Prentice Hall.
- Côté, I. (2012). *A systematic approach to software evolution*. Baden-Baden: Deutscher Wissenschafts-Verlag.
- Côté, I., Hatebur, D., Heisel, M., Schmidt, H., & Wentzlaff, I. (2008). A systematic account of problem frames. In *Proceedings of the European Conference on Pattern Languages of Programs (EuroPLoP)*. Universitätsverlag Konstanz.

- Côté, I., Hatebur, D., Heisel, M., & Schmidt, H. (2011). UML4PF—A tool for problem-oriented requirements analysis. In *Proceedings of the International Conference On Requirements Engineering (RE)* (pp. 349–350). IEEE Computer Society.
- Fabian, B., Gürses, S., Heisel, M., Santen, T., & Schmidt, H., (2010). A comparison of security requirements engineering methods. *Requirements Engineering—Special Issue on Security Requirements Engineering*, 15(1), 7–40.
- Hatebur, D. (2012). *Pattern and component-based development of dependable systems*. Deutscher Wissenschafts-Verlag (DWV) Baden-Baden.
- Hatebur, D., & Heisel, M. (2009). A foundation for requirements analysis of dependable software. In *Proceedings of the International Conference on Computer Safety, Reliability and Security (SAFECOMP)* (pp. 311–325). Springer.
- Hatebur, D., & Heisel, M. (2010). A UML profile for requirements analysis of dependable software. In *Proceedings of the International Conference on Computer Safety, Reliability and Security (SAFECOMP)* (pp. 317–331). Springer.
- Heisel, M. (1998). Agendas—A concept to guide software development activities. In *Proceedings of the IFIP TC2 WG2.4 Working Conference on Systems Implementation: Languages, Methods and Tools* (pp. 19–32). Chapman & Hall London.
- ISO. (2011). ISO 26262—Road Vehicles—Functional Safety. Geneva, Switzerland: International Organization for Standardization (ISO).
- ISO/IEC. (2000). ISO/IEC 61508 Functional safety of electrical/electronic/programmable electronic safety-relevant systems. Geneva, Switzerland: International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC).
- ISO/IEC. (2005). Information technology—Security techniques—Information security management systems—Requirements (ISO/IEC 27001). Geneva, Switzerland: International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC).
- ISO/IEC. (2008). Information technology—Security techniques—Information security risk management (ISO/IEC 27005). Geneva, Switzerland: International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC).
- ISO/IEC. (2009). Information technology—Security techniques—Information security management systems—Overview and Vocabulary (ISO/IEC 27000). Geneva, Switzerland: International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC).
- ISO/IEC. (2012). Common Criteria for Information Technology Security Evaluation (ISO/IEC 15408). Geneva, Switzerland: International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC).
- ISO/IEC. (2013). Information technology—Security techniques—Information security management systems—Requirements (ISO/IEC 27001). Geneva, Switzerland: International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC).
- ISO/IEC. (2014). Information technology—Security techniques—Information security management systems—Overview and Vocabulary (ISO/IEC 27000). Geneva, Switzerland: International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC).
- ISO. (2009). ISO 31000 risk management—Principles and guidelines Geneva. International Organization for Standardization (ISO): Switzerland.
- Jackson, M. (2001). *Problem frames: Analyzing and structuring software development problems*. Boston: Addison-Wesley.
- Jackson, M., & Zave, P. (1995). Deriving specifications from requirements: An example. In *Proceedings of the 17th International Conference on Software Engineering* (pp. 15–24). ACM.
- Karpati, P., Sindre, G., & Opdahl, A. L. (2011). Characterising and analysing security requirements modelling initiatives. In *Proceedings of the International Conference on Availability, Reliability and Security (ARES)* (pp. 710–715). IEEE Computer Society.
- Klipper, S. (2010). Information Security Risk Management MIT ISO/IEC 27005: Risikomanagement MIT ISO/IEC 27001, 27005 und 31010. Vieweg+Teubner.
- Lund, M. S., Solhaug, B., & Stølen, K. (2010). *Model-driven risk analysis: The CORAS approach* (1st ed.). London: Springer.



- Mahler, T. (2010). *Legal risk management*. Unpublished doctoral dissertation, University of Oslo.
- Massacci, F., Mylopoulos, J., & Zannone, N. (2010). Security requirements engineering: The SI\* modeling language and the secure tropos methodology. *Advances in Intelligent Information Systems*, 265, 147–174.
- UML Revision Task Force. (2010). OMG unified modeling language: Superstructure.

Pattern and Security Requirements  
Engineering-Based Establishment of Security Standards  
Beckers, K.  
2015, XXV, 474 p. 186 illus., Hardcover  
ISBN: 978-3-319-16663-6