A Meta-Pattern and Pattern Form For Context-Patterns

Kristian Beckers, paluno – The Ruhr Institute for Software Technology
Stephan Faßbender, paluno – The Ruhr Institute for Software Technology
Maritta Heisel, paluno – The Ruhr Institute for Software Technology

In a previous EuroPlop publication we introduced a catalog of context-patterns. We described common structures and stakeholders for several different domains in our context-patterns. The common elements of the context were obtained from observations about the domain in terms of standards, domain specific-publications, and implementations. Whenever a system-to-be is already described by a context-pattern, one can use this context-pattern to elicit domain knowledge via instantiation of the context-pattern. Moreover, we analyzed the common concepts in our context-patterns and created a meta-model to describe the relations between these concepts. This meta-model was the initial step towards a pattern language for context-patterns. In this work, we show the consequent next step for the definition of a pattern language for context-patterns.

Categories and Subject Descriptors: H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—*Evaluation/ methodology*; H.1.2 [**Models and Principles**]: User/Machine Systems—*Human Information Processing*; I.5.1 [**Pattern Recognition**]: Models—*Neural Nets*

General Terms: Human Factors

Additional Key Words and Phrases: Domain Knowledge, Requirements Engineering, Context Establishment

ACM Reference Format:

Beckers K., Faßbender S. 2014.A Pattern Language for Context-Patterns jn 2, 3, Article 1 (May 2010), 19 pages.

1. INTRODUCTION

The long known credo of requirements engineering states that it is challenging to build the right system if you do not know what right is. Fixing a defect when it is already fielded is reported to be up to eighty times more expensive than fixing the corresponding requirements defects early on [Boehm and Papaccio 1988; Willis 1998]. But eliciting good requirements is not an easy task [Firesmith 2003]. One important information for eliciting good requirements is domain knowledge which should be as complete as possible. Domain knowledge contains information about the involved stakeholders, the relations between the stakeholders, systems in the technical environment of the system-to-be and so forth. It is an open research question of *how* to elicit domain knowledge correctly for effective requirements elicitation [Niknafs and Berry 2012]. Fabian et al. [2010] concluded in their survey about these methods that it is not yet state of the art to consider domain knowledge.

We propose to built patterns for a structured domain knowledge elicitation. Our context-patterns support the structured elicitation of domain knowledge and we showed a number of these in the previous works of ours [Beckers et al. 2012; Beckers et al. 2012; Beckers and Faßbender 2012; Beckers et al. 2012; Faßbender and Heisel 2013].

Author's address: Kristian Beckers, Oststrasse 99, 47057 Duisburg, Germany; email kristian.beckers@uni-duisburg-essen.de; Stephan Faßbender, Oststrasse 99, 47057 Duisburg, Germany; email: stephan.fassbender@uni-due.de; Maritta Heisel, Oststrasse 99, 47057 Duisburg, Germany; email: stephan.fassbender@uni-due.de

However, these patterns share a common base in terms of a meta-model, which we presented in [Beckers et al. 2013]. This indicates, that the pattern share some commonalities on a more abstract level. In this paper, we try to describe a meta-process which identifies the commonalities in deriving and describing a context-pattern. Both, the meta-model and the meta-process are then used for forming a meta-pattern. We present the meta-pattern using a new pattern form for context-pattern. The meta pattern and the pattern form can help to describe newly found context-pattern, which we will show by using them for describing a new context-pattern.

In the next section, we summarize and describe the meta-model for context-patterns which we published in an earlier work of ours [Beckers et al. 2013]. Our meta-model shows common elements of context-patterns and their relations (Sec. 2). The meta-model only focuses on the structural commonalities between context-pattern. In this work, we present the behavioral commonalities in form of a meta-process in Sec. 3. The meta-model and the meta-process form the basis for the context meta-pattern. A form to represent this pattern and context-patterns is introduced in Sec. 4. Afterward, the context meta-pattern is shown in Sec. 5. Then, the application of the context-meta pattern for describing the smart grid context-pattern is shown in Sec. **??**. Finally, the work is concluded in Sec. 7.

2. A META-MODEL FOR CONTEXT-PATTERN

In this section, which is a summery of a previous work [Beckers et al. 2013], we present a meta-model for building context-patterns that consider domain knowledge during the analysis phase of software engineering. We consider different kinds of domain knowledge, e.g., technical domain knowledge. Therefore, we used a bottom-up approach, starting with a set of previously and independently developed context-patterns.

We identified the common concepts in our existing context-patterns [Beckers et al. 2012; Beckers et al. 2012; Beckers and Faßbender 2012; Beckers et al. 2012], and aggregate this knowledge into a meta-model of elements one has to talk and think about when describing a new context-pattern [Beckers et al. 2013].

This is quite similar to what Jackson [Jackson 2001] proposed for requirements. He defined a meta-model of reoccurring domains, like causal, biddable and lexical domains. These domains are used to define basic requirements patterns, so-called *Problem Frames* [Jackson 2001].

This meta-model has several benefits. First, it forms a uniform basis for our context-patterns, making them comparable. Second, findings and results for one pattern can be transferred to the other patterns via a generalization. Third, the meta-model contains the important conceptual elements for context-patterns. Fourth, it enables us to form a pattern language for context-patterns. However, in this work we focus on the aspects of the meta-model which are important for forming a meta-pattern.

Using this meta-model we empower requirements and software engineers to describe their own context-patterns, which capture the most important parts for understanding the context of a system-to-be. The meta-model was derived in a bottom-up way from the different patterns we described independently for different domains. For the process of deriving the general elements, which then form the meta-model, we started to analyze each context-pattern in isolation. For each element in a context-pattern we discussed what the general concept behind this element is or if it is a general concept in itself. In a next phase we harmonized the conceptual elements by comparing the found elements, merging them if needed and setting up their relations. This way we got a coherent set of conceptual elements over all patterns. In the last phase we had to choose which conceptual elements should be part of the meta-model. Finally, we formed the meta-model as depicted in Fig. 1 out of the selected conceptual elements. The meta-model was modeled using the UML notation.

The root element is the `Pattern` itself. Each pattern contains at least one `Area`. In general, an area contains elements of either a technical or organizational view. An area can contain other areas, which do not need to be of the same view. An area can concern either a `Machine`, i.e. the thing to be developed, or an `Environment`, which in turn contains elements that have some kind of relation to the machine, or a `Layer`, which encapsulates a set of elements.

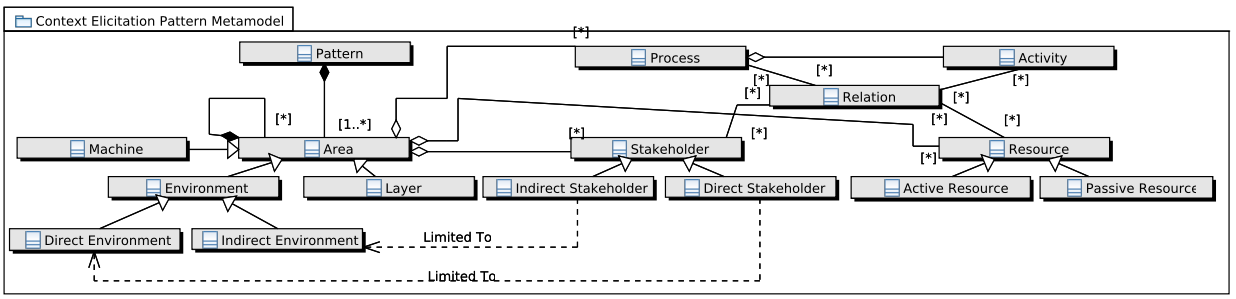

Fig. 1: Context-Pattern Meta-model

The environment can be further refined. There are elements which directly interact with the machine, captured in the `Direct Environment`. And there are elements which have an influence on the system via elements of the direct environment, captured by the `Indirect Environment`.

An element which is part of an `Area` can be a `Process`, a `Stakeholder`, or a `Resource`. A process describes some kind of workflow or sequence of activities. Therefore, it can contain `Activities`. A stakeholder describes a person, a group of persons, or organizational units, which have some kind of influence on the machine. A stakeholder can be refined to a `Direct Stakeholder` who interacts directly with the machine, and an `Indirect Stakeholder` who only interacts with direct stakeholders but has some interest in or influence on the machine. A `Resource` describes some physical or non physical (e.g., information) element which is needed to run the machine or which is processed by the machine and which is not a stakeholder. A resource can be an `Active Resource` with some behavior or a `Passive Resource` without any behavior. Several `Relations` are possible between processes, activities, stakeholders and resources. Sub-typing is not possible in this case, as the relations within domains turned out to be to diverse.

This meta-model has several benefits. First, it forms a uniform basis for our context-patterns, making them comparable. If a method already makes use of one of the patterns, it is now easy to generalize the usage to the elements of the meta-model. This enables one to replace a given used pattern by another one easily. Second, findings and results for one pattern can be transferred to the other pattern via a generalization to the meta-model elements. Third, the meta-model contains the important conceptual elements for context-patterns. Thus, it is helpful to know these elements and search for them in a specific domain when setting up a new context-pattern for a domain. Fourth, it enables to form a pattern language for the context-pattern. The common meta-model eases relating the patterns to each other.

3. A META-PROCESS FOR CONTEXT-PATTERN

The meta-model describes the common entities and relations of context-pattern. But it only describes the structural commonalities we identified in [Beckers et al. 2013] missing the common dynamics, namely the process we used to derive and describe the context-pattern. Hence, we mined protocols, generated artifacts, and intermediate results for commonalities between activities taken. We reflected the insights and compared it to our experience. The result is a meta-process (see Fig. 2).

The method is conducted by *pattern / IT experts* and *domain experts* (see Fig. 6). From our experience, it is not efficient to start the collaboration between the pattern and domain experts right from the beginning. The reason is, that without any common ground discussions are often fruitless. And the willingness of domain experts to collaborate, when one has nothing to provide as starting point, is very small. Additionally, the pattern experts can hardly judge or ask subsequent questions to the answers and results they obtain. As consequence, the

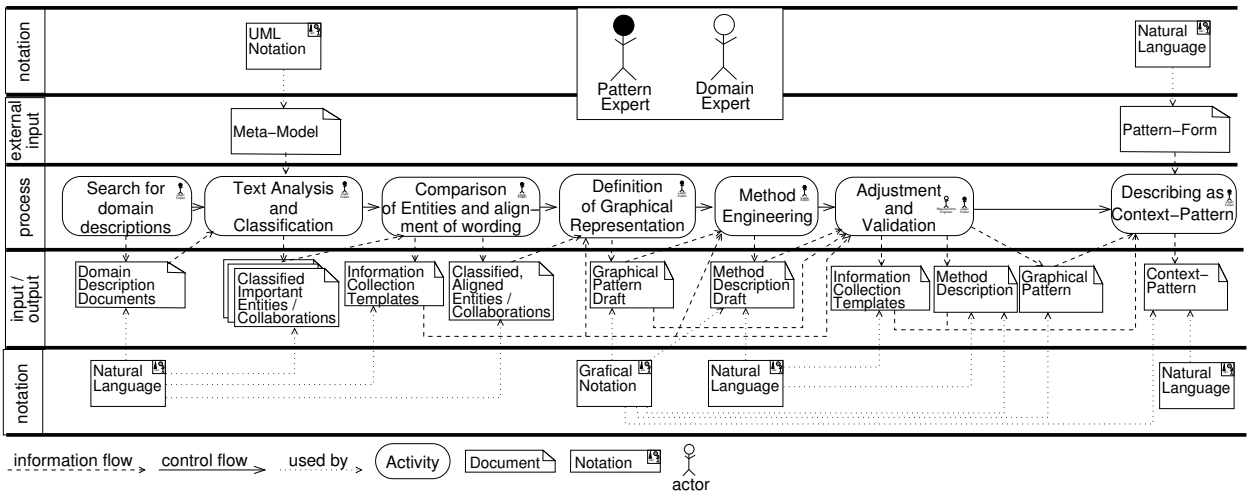

Fig. 2: A process for deriving and describing context pattern

pattern experts should get a basic understanding of the domain and derive preliminary results from freely available documents describing the domain.

Hence, the method starts with *searching for documents* describing the unknown domain. Such documents can be domain standards, technical documents, or scientific or white paper. There should be as many *domain description documents*, which are central for a domain and of good quality, as possible but at least 3 for the subsequent steps .

Once these documents are collected, the text and contained figures have to be analyzed for important entities. At this point, the *meta-model* can guide the pattern expert. Reoccurring terms can be collected and mapped to the meta-model. This way, a first impression of the basic semantic of a term is captured and entities are formed. Moreover, the meta-model gives guidance for which terms one should search.

After all entities and relations are collected from the documents in isolation, the *classified important entities / relations* have to be *compared and the wording needs to be aligned*. Entities with the same semantic are group and one specific term to name them is selected. This way, a coherent set of *classified and aligned Entities and collaborations* is derived from the different documents describing the domain under investigation. While classifying the entities, one should also track which information is given about the entities in the source documents. Information, which is always given or requested for an entity in all source documents is described using, so called, *information collection templates*. These templates serve later on, when the context-pattern is used for context elicitation, as kind of questionnaire to collect information about an entity. We also tried to discover the commonalities between the different information collection templates. Only for stakeholder entities we were able to find and describe them (see Table I). All other entities are to diverse to describe a shared template beside name and description.

Next, the classified and aligned entities and collaborations between them are *expressed and visualized using a graphical representation*. The graphical representation should only capture the most important information about a domain to keep the graphical representation understandable. All other information should be store in the according information collection templates. From our experience, the graphical representation is the central mean to support communication, discussion, and information collection. Iterating over the graphic and discussing all entities and collaborations, visualized as relations, is really helpful to get a common ground and understanding in the first place. Additionally, the graphic is one mean to assure completeness in the sense that no entity is forgotten in the end.

Table I. : Information Collection Template Pattern for Stakeholders

Name. What is the name or identifier of the stakeholder?

Description. Which important properties does the stakeholder have? What characterizes the stakeholder? What is its place in the environment?

Motivation. Which objectives does the stakeholder follow? Why does the stakeholder influence the organization(s) / provider(s)?

Top Level Goals. Which top level goals does the stakeholder have?
☐ **Adaptability** ☐ **Compliance** ☐ **Economy** ☐ **Efficiency** ☐ **Evolvability** ☐ **Learnability** ☐ **Maintainability** ☐ **Modularity** ☐ **Performance** ☐ **Portability** ☐ **Privacy** ☐ **Reliability** ☐ **Resilience** ☐ **Re-Usability** ☐ **Robustness** ☐ **Safety** ☐ **Scalability** ☐ **Security** ☐ **Testability** ☐ **Understandability** ☐ **Usability**

Kind.
☐ **Specific** Is the stakeholder a real entity? Is the stakeholder not used to represent a group?
☐ **Representative** Is the stakeholder a real existing entity? Is this stakeholder used as proxy for a group of homogeneous stakeholders?
☐ **Group** Is the stakeholder not a real existing entity? Is this stakeholder used to describe for a group of homogeneous stakeholders?
☐ **Role** Can this stakeholder be shared through groups of heterogeneous stakeholders? Are there well-defined rights and permissions for this stakeholder?

And we noticed, that discussion are more structured and less abstract when people have something to point and work with. For the graphical representation, we use the UML [UML Revision Task Force 2012] or UMLish diagrams. The reason is, that many people are familiar with this notation. For us, the UML was a reasonable choice, but one may select what ever suits him / her best. For example, when working with the US-military one would probably select a IDEF-like notation (e.g. IDEF1 [National Institute of Standards and Technology 2014a] or IDEF4 [National Institute of Standards and Technology 2014b]). In the end, one should have *graphical pattern draft* regardless which notation is used.

The next activity, *method engineering*, is also of great importance. In the beginning, we used context-patterns without a proper structured method. As consequence the process of context elicitation was difficult, as we sometimes were missing information for an entity at hand, which we should have collected before hand. Or information was collected several times. And the coherence of the collected information was hard to maintain. To solve these problems, we started to define methods for collecting the information. We use the agenda principle as described in [Heisel 1998]. A agenda describes a sequence of activities, with necessary inputs, the desired outputs, actors, the notations used, and validation conditions to check coherence. We accompanied the purely textual representation using agenda diagrams. Figure 2 is an instance of such an diagram. One can use what ever he / she wants to describe processes. For us the agenda principle turned out to be useful. The output of the activity method engineering is a *method description draft*.

All drafts should be *adjusted and validated* in discussions with domain experts. Up to this point, the pattern experts only worked on the basis of documents. Doing so introduces the threats of misunderstanding, incompleteness, and biased results due to the influence of the pattern experts. To discover such issues, it is indispensable to get a review by domain experts. But as the there is already a common ground to discuss and inputs to be reviewed conducting this activity is not an big issue, nevertheless some iterations might be needed. The final versions of the *information collection templates*, the *method description*, and the *graphical pattern* are the result of this activity.

Last, all final results are compiled to a context-pattern description using a *pattern form*.

4. A PATTERN FORM FOR CONTEXT-PATTERN

Our pattern form is adaptation of different existing pattern forms [Fowler 1996; 2002; Schumacher et al. 2006; Gamma et al. 1994]. Our pattern form is shown and explained in the following:

Summary	**Pattern Name**	A unique and descriptive name	**Classification**	The type of pattern. For context-pattern there are 4 types possible: *Meta-Pattern.* describing the essence and commonalities of a context-pattern family *Technical.* A context-pattern which focuses on the technical view solely. *Organizational.* A context-pattern which focuses on the organizational view solely. *Organizational & Technical.* A context-pattern which combines an organizational and technical view.
	Related Patterns	Which patterns are related to the pattern at hand?		
	Intent	Short description when and why to use the pattern.		
	Known Uses	List of examples where the pattern was applied and from which the pattern was derived.		
Motivation	**Context**	A description of the context where the problem, which the pattern solves, might occur.		
	Problem	A description of the problem.		
	Forces	A description of the forces which influence the problem and solution.		
	Example	A short narrative which exemplifies the context, problem and forces.		
Solution	**Structure**	A grafical representation of the pattern in a suitable notation		

Solution	Entities	Which entities are part of the pattern? A description of those entities.
	Collaborations	What relations between entities exist in the pattern? A description of relations.
	Method	A description how to use the pattern.

5. A META-PATTERN FOR CONTEXT-PATTERN

The meta-model (see Sec. 2) describes the common entities and relations of context-pattern. The meta-process (see Sec. 3) describes the commonalities in deriving and describing a context-pattern. The meta-process and the meta-model form the solution to our the context meta-pattern, which is described in the following using our pattern form:

	Pattern Name	Context Meta-Pattern **Classification** Meta-Pattern
	Related Patterns	-
Summary	Intent	This pattern is used whenever one has to elicit context information for a domain and no context-pattern for this domain is available.
	Known Uses	—Cloud System Analysis pattern [Beckers et al. 2011; Beckers et al. 2012] —Law pattern [Beckers et al. 2012; Faßbender and Heisel 2013] —Law identification pattern [Beckers et al. 2012; Faßbender and Heisel 2013] —P2P pattern [Beckers and Faßbender 2012] —SOA Layer pattern [Beckers et al. 2012] —SOA Layer Stakeholder pattern [Beckers et al. 2012]
Motivation	Context	Context information elicitation is a general problem when dealing with IT systems. Often the IT experts who have to analyze, asses, implement, run, or maintain a system are not experts in the domain(s) where the system is used. Moreover, even when they know the domain, big parts of the required knowledge is tacit knowledge within an organization. Hence, they do not posses the detailed knowledge about the environment of the IT system. And for a complex IT system the environment is also very complex and contains diverse legacy systems, people interacting with the system, people influencing the system and so forth. All these elements bring in own constraints, goals, knowledge and so forth.
	Problem	The general problem is centered around different questions: (1) Which information has to be collected in a domain which is unknown? (2) How to represent and persist the important elements of an domain found while analyzing the domain for (re)use? (3) How to collect knowledge about the important elements in structured way especially when it is tacit knowledge? (4) How to externalize and store the collected information that it is useful afterward?

Motivation

Forces

There are several factors which have an influence on the context elicitation for a making it complicated to find a solution:

No coherent and widely accepted definition of a domain available.. For many domains there is not one central definition. In most cases there are several commonly accepted descriptions of domain such as scientific publications, standards, regulations, surveys and so forth. Those sources differ in some points as they often have a different view on a domain or use a different wording. The commonalities, and therefore main elements, in these documents have to be discovered and the wording needs to be aligned.

Knowledge about important elements is scattered, diverse, and tacit within an organization.. For collecting the information one needs to know where to collect it. In many cases, this is the first challenge as there are several places and people where this information resides. Again the wordings is often not aligned, hence, a mapping is needed. And the different persons have a different view on the important elements. Thus, the view of all those people needs to be harmonized. And their tacit knowledge needs to be externalized and connected. All these things make a communication-heavy process necessary. For this process, means which support and structure the communication are needed.

Example

One application domain of the NESSoS project, in which we take part in, is the smart grid. Our task within this project is to deliver solutions to analyze the smart grid regarding security, privacy and compliance, for example with standards, in the early phase of the software development life cycle. Our industrial partner for this task is Siemens. While we are experienced in requirements engineering, security, privacy and compliance in general, we did not have any further knowledge about smart grids. Hence, the results of applying our methods were unsatisfactory in the beginning. The main problem was, that we did not talk the language of Siemens and Siemens did not understand what we need to know and how to describe this knowledge. As we also did not know what the important parts of a smart grid are, we could not ask the right questions. The result was a slow, trail and error process annoying both sides. We had to change something to cope with this situation.

Solution

Structure

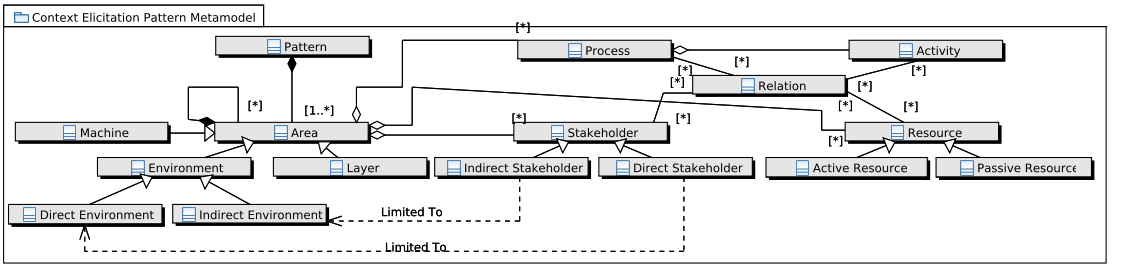

Fig. 3: Important entities and relations for the Context Meta-Pattern

Table IV Information Collection Template Pattern for Stakeholders

Name. What is the name or identifier of the stakeholder?

Description. Which important properties does the stakeholder have? What characterizes the stakeholder? What is its place in the environment?

Motivation. Which objectives does the stakeholder follow? Why does the stakeholder influence the organization(s) / provider(s)?

Top Level Goals. Which top level goals does the stakeholder have?
☐ **Adaptability** ☐ **Compliance** ☐ **Economy** ☐ **Efficiency** ☐ **Evolvability** ☐ **Learnability**
☐ **Maintainability** ☐ **Modularity** ☐ **Performance** ☐ **Portability** ☐ **Privacy** ☐ **Reliability** ☐ **Resilience**
☐ **Re-Usability** ☐ **Robustness** ☐ **Safety** ☐ **Scalability** ☐ **Security** ☐ **Testability**
☐ **Understandability** ☐ **Usability**

Kind.
☐ **Specific** Is the stakeholder a real entity? Is the stakeholder not used to represent a group?
☐ **Representative** Is the stakeholder a real existing entity? Is this stakeholder used as proxy for a group of homogeneous stakeholders?
☐ **Group** Is the stakeholder not a real existing entity? Is this stakeholder used to describe for a group of homogeneous stakeholders?
☐ **Role** Can this stakeholder be shared through groups of heterogeneous stakeholders? Are there well-defined rights and permissions for this stakeholder?

Solution

Entities

Area. Each pattern contains at least one `Area`. In general, an area contains elements of either a technical or organizational view. An area can contain other areas, which do not need to be of the same view.

Machine. Area concerned with the `Machine`, i.e. the thing to be developed

Environment. The `Environment` is an area, which in turn contains elements that have some kind of relation to the machine.

Direct Environment. There are elements which directly interact with the machine, captured in the `Direct Environment`.

Indirect Environment. There are elements which have an influence on the system via elements of the direct environment, captured by the `Indirect Environment`

Layer. A `Layer` is an area, which encapsulates a set of elements within the environment or a machine.

Process. A `Process` describes some kind of workflow or sequence of activities.

Activities. `Activities` are a part of a process.

Stakeholder. A `Stakeholder` describes a person, a group of persons, or organizational units, which have some kind of influence on the machine.

Direct Stakeholder. `Direct Stakeholder` interacts directly with the machine.

Indirect Stakeholder. `Indirect Stakeholder` who only interacts with direct stakeholders but has some interest in or influence on the machine.

Resource. A `Resource` describes some physical or non physical (e.g., information) element which is needed to run the machine or which is processed by the machine and which is not a stakeholder.

Active Resource. A resource can be an `Active Resource` with some behavior.

Passive Resource. A resource can be a `Passive Resource` without any behavior.

Collaborations

Relation. Several `Relations` are possible between processes, activities, stakeholders and resources.

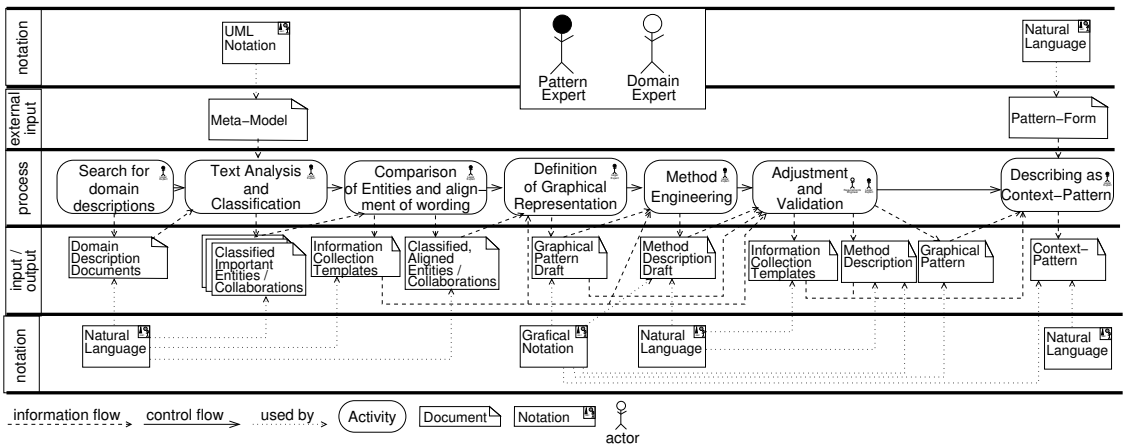

Fig. 6: Process Pattern for Using the Meta-Model to derive a context pattern.

Solution

Method

The method is conducted by *pattern / IT experts* and *domain experts* (see Fig. 6). The method starts with *searching for documents* describing the unknown domain. Such documents can be domain standards, technical documents, or scientific or white paper. There should be as many *domain description documents*, which are central for a domain and of good quality, as possible but at least 3 for the subsequent steps . Once these documents are collected, the text and contained figures have to be analyzed for important entities. At this point, the *meta-model* can guide the pattern expert. Reoccurring terms can be collected and mapped to the meta-model. This way, a first impression of the basic semantic of a term is captured and entities are formed. Moreover, the meta-model gives guidance for which terms one should search. After all entities and relations are collected from the documents in isolation, the *classified important entities / relations* have to be *compared and the wording needs to be aligned*. Entities with the same semantic are group and one specific term to name them is selected. This way, a coherent set of *classified and aligned Entities and collaborations* is derived from the different documents describing the domain under investigation. While classifying the entities, one should also track which information is given about the entities in the source documents. Information, which is always given or requested for an entity in all source documents is described using, so called, *information collection templates*. These templates serve later on, when the context-pattern is used for context elicitation, as kind of questionnaire to collect information about an entity. The common information collection template for stakeholder is shown in Table IV. Next, the classified and aligned entities and collaborations between them are *expressed and visualized using a graphical representation*. The graphical representation should only capture the most important information about a domain to keep the graphical representation understandable. All other information should be store in the according information collection templates. In the end, one should have *graphical pattern draft* regardless which notation is used. The next activity, *method engineering*. For this activity, We use the agenda principle as described in [Heisel 1998]. A agenda describes a sequence of activities, with necessary inputs, the desired outputs, actors, the notations used, and validation conditions to check coherence. We accompanied the purely textual representation using agenda diagrams. Figure 2 is an instance of such an diagram. One can use what ever he / she wants to describe processes. For us the agenda principle turned out to be useful. The output of the activity method engineering is a *method description draft*. All drafts should be *adjusted and validated* in discussions with domain experts. The final versions of the *information collection templates*, the *method description*, and the *graphical pattern* are the result of this activity. Last, all final results are compiled to a context-pattern description using a *pattern form*.

6. THE SMART GRID CONTEXT-PATTERN

	Pattern Name	Smart Grid Context-Pattern	**Classification**	Organizational & Technical
	Related Patterns	Cloud System Analysis Pattern		
Summary	**Intent**	This pattern can be used to elicit the context of a application or system which is part of a smart grid.		
	Known Uses	—CC protection profiles for Smart Meters [BSI 2011; 2013] —The documentation of the OpenNode project [OPEN node project 2010; 2011] —The documentation of the OpenMeter project [OPEN meter project 2009] —The industry case studies from the NESSoS project —The British Smart Grid implementation program [Department of Energy and Climate Change 2011b; 2011a]		
Motivation	**Context**	Deriving from the definitions of the European Commission [Commission of the European communities 2011], the European Smart Grid Task Force[1], and the Office of Electricity Transmission and Distribution[2], the Smart Grid can be described as a large, flexible, self-monitoring, auto-balancing, and self-regulating electricity infrastructure which uses two-way digital communication to gather and respond on information in an automated manner in order to improve the efficiency, reliability (meaning safety and security), and sustainability of the production and distribution of energy. This new infrastructure will be able to efficiently integrate the behavior and actions of all users connected to it. This means generators, consumers, those that do both and other third parties that provide services outside of energy generation.		
	Problem	The general problem of describing the context within a smart grid is centered around different questions: (1) Which information has to be collected to describe the context of an application or system which will be part of a smart grid? (2) How to collect knowledge about the important elements of an smart grid in structured way? (3) How to externalize and store the collected information that it is useful afterward?		

[1] `http://ec.europa.eu/energy/gas_electricity/smartgrids/taskforce_en.htm` (last visited on 15-12-2013)
[2] `http://energy.gov/oe/technology-development/smart-grid` (last visited on 15-12-2013)

Motivation

Forces

There are several factors which have an influence on the context elicitation for a making it complicated to find a solution:

No coherent and widely accepted definition of a domain available.. For many domains there is not one central definition. In most cases there are several commonly accepted descriptions of domain such as scientific publications, standards, regulations, surveys and so forth. Those sources differ in some points as they often have a different view on a domain or use a different wording. The commonalities, and therefore main elements, in these documents have to be discovered and the wording needs to be aligned.

Knowledge about important elements is scattered, diverse, and tacit within an organization.. For collecting the information one needs to know where to collect it. In many cases, this is the first challenge as there are several places and people where this information resides. Again the wordings is often not aligned, hence, a mapping is needed. And the different persons have a different view on the important elements. Thus, the view of all those people needs to be harmonized. And their tacit knowledge needs to be externalized and connected. All these things make a communication-heavy process necessary. For this process, means which support and structure the communication are needed.

Example

One application domain of the NESSoS project, in which we take part in, is the smart grid. Our task within this project is to deliver solutions to analyze the smart grid regarding security, privacy and compliance, for example with standards, in the early phase of the software development life cycle. Our industrial partner for this task is Siemens. While we are experienced in requirements engineering, security, privacy and compliance in general, we did not have any further knowledge about smart grids. Hence, the results of applying our methods were unsatisfactory in the beginning. The main problem was, that we did not talk the language of Siemens and Siemens did not understand what we need to know and how to describe this knowledge. As we also did not know what the important parts of a smart grid are, we could not ask the right questions. The result was a slow, trail and error process annoying both sides. We had to change something to cope with this situation.

Solution

Structure

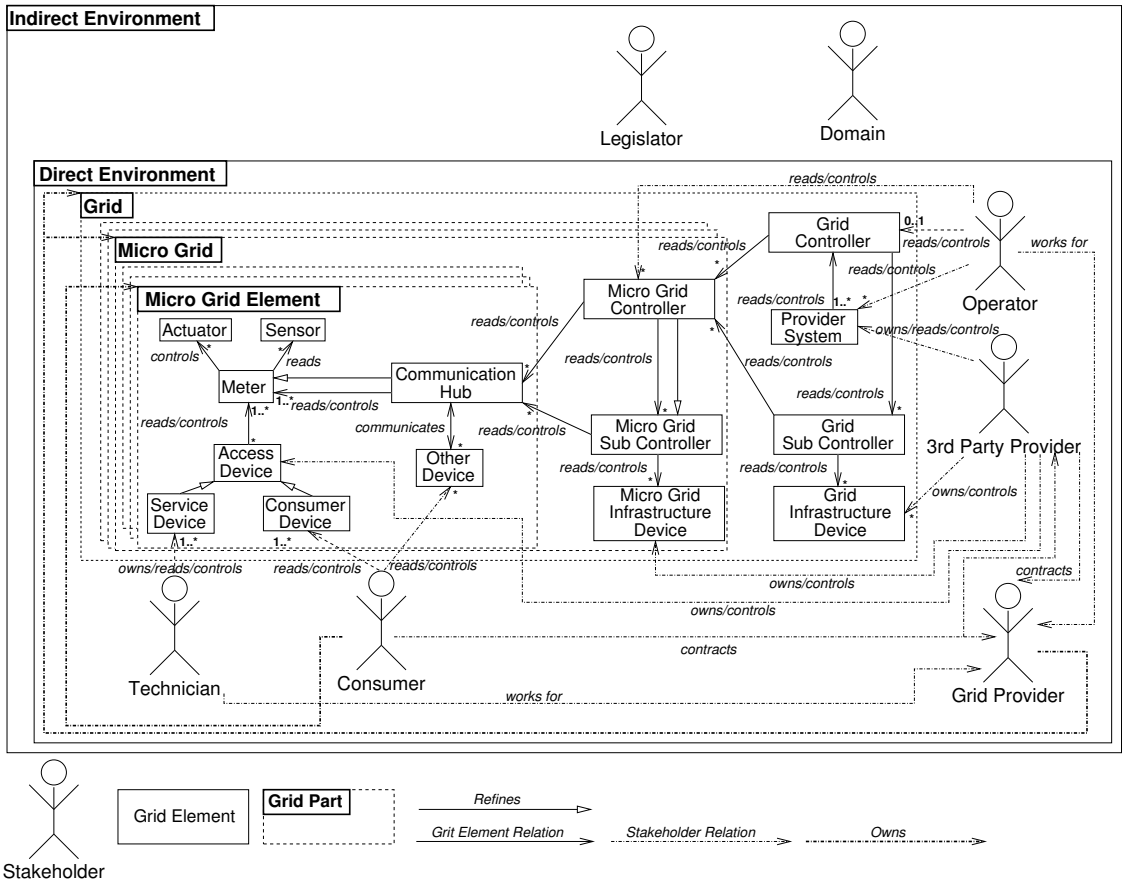

Fig. 5: Important entities and relations for the Context Meta-Pattern

Table VI Information Collection Template for Resources

Name. What is the name or identifier of the resource? *Description.* Which important properties does the respource have? What characterizes the resource? What is its place in the environment?

Table VII Information Collection Template for Direct Stakeholders

Name. What is the name or identifier of the stakeholder?

Description. Which important properties does the stakeholder have? What characterizes the stakeholder? What is its place in the environment?

Motivation. Which objectives does the stakeholder follow? Why does the stakeholder influence the organization(s) / provider(s)?

Top Level Goals. Which top level goals does the stakeholder have?
☐ **Adaptability** ☐ **Compliance** ☐ **Economy** ☐ **Efficiency** ☐ **Evolvability** ☐ **Learnability**
☐ **Maintainability** ☐ **Modularity** ☐ **Performance** ☐ **Portability** ☐ **Privacy** ☐ **Reliability** ☐ **Resilience**
☐ **Re-Usability** ☐ **Robustness** ☐ **Safety** ☐ **Scalability** ☐ **Security** ☐ **Testability**
☐ **Understandability** ☐ **Usability**

Kind.
☐ **Specific** Is the stakeholder a real entity? Is the stakeholder not used to represent a group?
☐ **Representative** Is the stakeholder a real existing entity? Is this stakeholder used as proxy for a group of homogeneous stakeholders?
☐ **Group** Is the stakeholder not a real existing entity? Is this stakeholder used to describe for a group of homogeneous stakeholders?
☐ **Role** Can this stakeholder be shared through groups of heterogeneous stakeholders? Are there well-defined rights and permissions for this stakeholder?

Table VIII Information Collection Template for Indirect Stakeholders

Name. What is the name or identifier of the stakeholder?

Description. Which important properties does the stakeholder have? What characterizes the stakeholder? What is its place in the environment?

Motivation. Which objectives does the stakeholder follow? Why does the stakeholder influence the organization(s) / provider(s)?

Top Level Goals. Which top level goals does the stakeholder have?
☐ **Adaptability** ☐ **Compliance** ☐ **Economy** ☐ **Efficiency** ☐ **Evolvability** ☐ **Learnability**
☐ **Maintainability** ☐ **Modularity** ☐ **Performance** ☐ **Portability** ☐ **Privacy** ☐ **Reliability** ☐ **Resilience**
☐ **Re-Usability** ☐ **Robustness** ☐ **Safety** ☐ **Scalability** ☐ **Security** ☐ **Testability**
☐ **Understandability** ☐ **Usability**

Kind.
☐ **Specific** Is the stakeholder a real entity? Is the stakeholder not used to represent a group?
☐ **Representative** Is the stakeholder a real existing entity? Is this stakeholder used as proxy for a group of homogeneous stakeholders?
☐ **Group** Is the stakeholder not a real existing entity? Is this stakeholder used to describe for a group of homogeneous stakeholders?
☐ **Role** Can this stakeholder be shared through groups of heterogeneous stakeholders? Are there well-defined rights and permissions for this stakeholder?

Influence.

On	Description	Severity
Which direct stakeholder is influenced?	Which kind of influence? What kind of enforcement? What is the base for the influence?	What is the rating for the severity of the influence?

optional entries

Law candidates. (Legislator) Which laws which might be of relevance for the actual grid (part) to be developed?

Domain-specific regulations. (Domain) Which domain-specific regulations including, for example, standards and best practice do exist?

Solution

Entities

Area

Grid. The grid is the thing to be build. This does not necessarily mean that all parts of the grid are object of an development project, but at least one will be the machine to be build.

Micro Grid.

Micro Grid Element.

Indirect Environment

Indirect Environment. The indirect stakeholders as part of the indirect environment have no influence and, in most cases, also no interest on the machine itself. But they have an influence on the direct stakeholders and therefore they are important for the system-to-be.

Direct Environment

Direct Environment. The direct environment contains all the direct stakeholders, who have a direct relation to one or more parts of the grid. Hence, they are able to directly influence the grid.

Indirect Stakeholder

Legislator. The legislator describes the government of a country for example. A legislator enacts and enforces different regulations which the system-to-be has to be compliant to.

Domain. The domain represents the special domains for the the system-to-be is developed. The domains influence is based on the self regulations of a domain, standards for this domain and so forth.

Direct Stakeholder

Operator. There are operators for different purposes, e.g. maintenance or billing. Operators which work for the the grid provider.

3rd Party Provider. Third party provider offer goods or services, which are delivered using the smart grid or are related to the smart grid. The third party providers also have a contractual relation to the grid provider.

Grid Provider. The grid provider owns and operates the grid and its major parts.

Consumer. The consumers have a contractual relation to the grid provider and consume, for example, energy which is provided by the grid or 3rd party providers.

Technician. The technicians work for the grid provider installing and maintaining the devices at the consumer side.

Solution **Entities**

Active Resource

Grid Controller.
Provider System.
Grid Sub Controller.
Grid Infrastructure Device.
Micro Grid Controller.
Micro Grid Sub Controller.
Micro Grid Infrastructure Device.
Communication Hub.
Other Device.
Meter.
Actuator.
Sensor.
Access Device.
Service Device.
Consumer Device.

Collaborations

Relation.

Solution **Method**

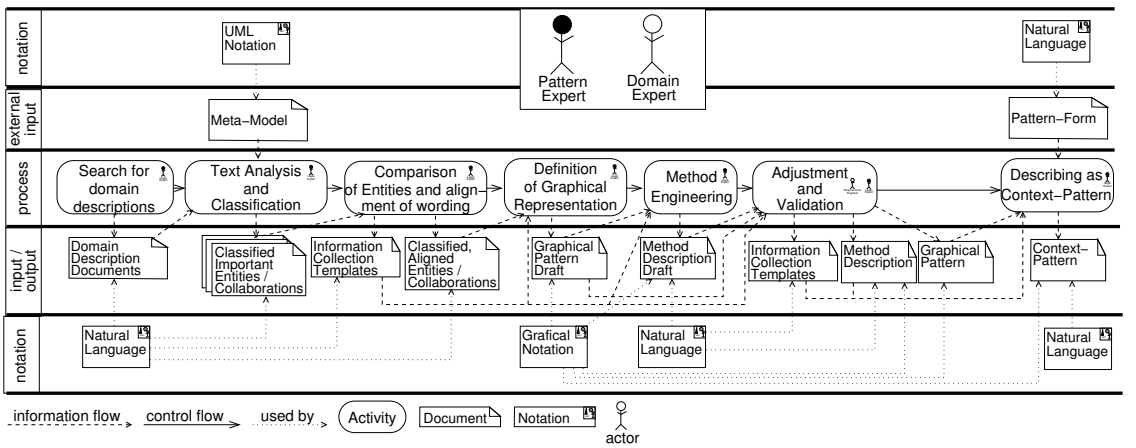

Fig. 6: Process Pattern for Using the Meta-Model to derive a context pattern.

7. CONCLUSION

In this work, we have presented the context meta-pattern. To form the context meta-pattern, we described the meta-model, which was the result of a previous work of ours, and accompanied it with a meta-process for deriving and describing context-pattern. The meta-model and meta-process are the core parts of the solution of our context meta-pattern. The complete context meta-pattern was then described using a new pattern form for context-pattern.

We have shown applicability of the context meta-pattern using the smart grid domain as domain to be described in the form of a context-pattern. The contribution of this work are as follows:

—A structured and guided meta-process for deriving and describing context-pattern.

—A pattern form for context-pattern.

—The context meta-pattern.

—The smart grid context-pattern.

The main future work is concerned with the activities *text analysis and classification* and *adjustment and validation*. These activities are central to the meta-process and complex. But currently there is no detailed description how to conduct them. To cope with this issue we strife for finding and describing scientific methods, which can support the actions taken for these two activities.

Another activity of ours, is to set up a pattern repository for context pattern and supporting information material about context-pattern.

8. ACKNOWLEDGEMENTS

This research was partially supported by the EU project Network of Excellence on Engineering Secure Future Internet Software Services and Systems (NESSoS, ICT-2009.1.4 Trustworthy ICT, Grant No. 256980), the German Research Foundation (DFG) under grant number HE3322/4-2, and the Ministry of Innovation, Science, Research and Technology of the German State of North Rhine-Westphalia and EFRE (Grant No. 300266902 and Grant No. 300267002).

REFERENCES

BECKERS, K. AND FASSBENDER, S. 2012. Peer-to-peer driven software engineering considering security, reliability, and performance. In *Proceedings of the International Conference on Availability, Reliability and Security (ARES) - 2nd International Workshop on Resilience and IT-Risk in Social Infrastructures (RISI 2012)*. IEEE Computer Society, 485–494.

BECKERS, K., FASSBENDER, S., AND HEISEL, M. 2013. A meta-model for context-patterns. In *Proceedings of the 18th European Conference on Pattern Languages of Programs (Europlop)*. ACM, –. Accepted for Publication.

BECKERS, K., FASSBENDER, S., HEISEL, M., AND MEIS, R. 2012. Pattern-based context establishment for service-oriented architectures. In *Software Service and Application Engineering*. LNCS 7365. Springer, 81–101.

BECKERS, K., FASSBENDER, S., KÜSTER, J.-C., AND SCHMIDT, H. 2012. A pattern-based method for identifying and analysing laws in the field of cloud computing compliance. In *Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ)*. LNCS Series, vol. 7195. Springer, 256–262.

BECKERS, K., FASSBENDER, S., AND SCHMIDT, H. 2012. An integrated method for pattern-based elicitation of legal requirements applied to a cloud computing example. In *Proceedings of the International Conference on Availability, Reliability and Security (ARES) - 2nd International Workshop on Resilience and IT-Risk in Social Infrastructures(RISI 2012)*. IEEE Computer Society, 463–472.

BECKERS, K., KÜSTER, J.-C., FASSBENDER, S., AND SCHMIDT, H. 2011. Pattern-based support for context establishment and asset identification of the ISO 27000 in the field of cloud computing. In *Proceedings of the International Conference on Availability, Reliability and Security (ARES)*. IEEE Computer Society, 327–333.

BOEHM, B. W. AND PAPACCIO, P. N. 1988. Understanding and controlling software costs. *IEEE Transactions on Software Engineering 14,* 10, 1462–1477.

BSI. 2011. Protection Profile for the Gateway of a Smart Metering System (Gateway PP). Version 01.01.01(final draft), Bundesamt für Sicherheit in der Informationstechnik (BSI) - Federal Office for Information Security Germany, Bonn,Germany. `https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/SmartMeter/PP-SmartMeter.pdf?__blob=publicationFile`.

BSI. 2013. Protection Profile for the Security Module of a Smart Meter Gateway (Security Module PP). Version 1.0), Bundesamt für Sicherheit in der Informationstechnik (BSI) - Federal Office for Information Security Germany, Bonn,Germany. `https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/SmartMeter/PP_Security_\%20Module.pdf?__blob=publicationFile`.

COMMISSION OF THE EUROPEAN COMMUNITIES. 2011. Communication from the commission to the european parliament, the council, the european economic and social committee and the committee of the regions.

DEPARTMENT OF ENERGY AND CLIMATE CHANGE. 2011a. Smart Metering Implementation Programme, Response to Prospectus Consultation, Design Requirements. Tech. rep., Office of Gas and Electricity Markets.

DEPARTMENT OF ENERGY AND CLIMATE CHANGE. 2011b. Smart Metering Implementation Programme, Response to Prospectus Consultation, Overview Document. Tech. rep., Office of Gas and Electricity Markets.

FABIAN, B., GÜRSES, S., HEISEL, M., SANTEN, T., AND SCHMIDT, H. 2010. A comparison of security requirements engineering methods. *Requirements Engineering – Special Issue on Security Requirements Engineering 15,* 1, 7–40.

FASSBENDER, S. AND HEISEL, M. 2013. From problems to laws in requirements engineering using model-transformation (best students paper award). In *ICSOFT 2013 - Proc. of the 8th Int. Conf. on Software Paradigm Trends*. INSTICC, SciTePress, 447–458.

FIRESMITH, D. 2003. Specifying good requirements. *Journal of Object Technology 2,* 4.

FOWLER, M. 1996. *Analysis Patterns: Reusable Object Models*. Addison-Wesley.

FOWLER, M. 2002. *Patterns of Enterprise Application Architecture*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

GAMMA, E., HELM, R., JOHNSON, R., AND VLISSIDES, J. 1994. *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley.

HEISEL, M. 1998. Agendas – a concept to guide software development activites. In *Proc. Systems Implementation 2000*. Chapman & Hall London, 19–32.

JACKSON, M. 2001. *Problem Frames. Analyzing and structuring software development problems*. Addison-Wesley.

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. 2014a. Idef1 information modelling. http://www.idef.com/pdf/IDEF1MR-part1.pdf, http://www.idef.com/pdf/IDEF1MR-part2.pdf.

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. 2014b. IDEF4 object-oriented design method. http://www.idef.com/pdf/Idef4.pdf.

NIKNAFS, A. AND BERRY, D. M. 2012. The impact of domain knowledge on the effectiveness of requirements idea generation during requirements elicitation. In *Requirements Engineering Conference (RE), 2012 20th IEEE International*. 181 –190.

OPEN METER PROJECT. 2009. D1.1 Requirements of AMI. Tech. rep., OPEN meter project.

OPEN NODE PROJECT. 2010. Evaluation of general requirements according state of the art . Tech. rep., OPEN node project.

OPEN NODE PROJECT. 2011. Functional Use cases. Tech. rep., OPEN node project.

SCHUMACHER, M., FERNANDEZ-BUGLIONI, E., HYBERTSON, D., BUSCHMANN, F., AND SOMMERLAD, P. 2006. *Security Patterns: Integrating Security and Systems Engineering*. Wiley.

UML REVISION TASK FORCE. 2012. *OMG Unified Modeling Language: Superstructure*. `http://www.omg.org/spec/UML/2.4/`.

WILLIS, R. 1998. *Hughes Aircraft's Widespread Deployment of a Continuously Improving Software Process*. AD-a358 993. Carnegie-mellon university.