

Editorial Manager(tm) for Requirements Engineering
Manuscript Draft

Manuscript Number:

Title: Enhancing Security Requirements Engineering by Organisational Learning

Article Type: S.I.: REFSQ 2011

Keywords: secure software engineering; requirements analysis; organisational learning; requirements workflow modeling

Corresponding Author: Eric Knauss

Corresponding Author's Institution: Leibniz Universität Hannover

First Author: Kurt Schneider, Prof. Dr.

Order of Authors: Kurt Schneider, Prof. Dr.;Eric Knauss;Siv Houmb, Dr.;Shareeful Islam, Dr.;Jan Jürjens, Prof. Dr.

Enhancing Security Requirements Engineering by Organisational Learning

Kurt Schneider and Eric Knauss

Software Engineering Group, Leibniz Universität Hannover, Welfengarten 1, 30167 Hannover, Germany,
E-Mail: {kurt.schneider, eric.knauss}@inf.uni-hannover.de

Siv Houmb

Secure-NOK AS, Norway, E-Mail: sivhoumb@securenok.com

Shareeful Islam

School of Computing, IT and Engineering, University of East London, 4-6 University way, London E16 2RD,
United Kingdom, E-Mail: shareeful@uel.ac.uk

Jan Jürjens

Chair for Software Engineering, TU Dortmund and Fraunhofer ISST, Baroper Strasse 301, 44227
Dortmund, Germany, E-Mail: <http://jan.jurjens.de>

Abstract. More and more software projects today are security-related in one way or the other. Requirements engineers often fail to recognise indicators for security problems which is a major source of security problems in practice. Identifying security-relevant requirements is labour-intensive and error-prone. In order to facilitate the security requirements elicitation process, we present an approach supporting organisational learning on security requirements by establishing company-wide experience resources, and a socio-technical network to benefit from them. The approach is based on modelling the flow of requirements and related experiences. Based on those models, we enable people to exchange experiences about security-requirements while they write and discuss project requirements. At the same time, the approach enables participating stakeholders to learn while they write requirements. This can increase security awareness and facilitate learning on both individual and organisational levels. As a basis for our approach, we introduce heuristic assistant tools which support reuse of existing security-related experiences. In particular, they include Bayesian classifiers which issue a warning automatically when new requirements seem to be security-relevant. Our results indicate that this is feasible, in particular if the classifier is trained with domain specific data and documents from previous projects. We show how the ability to identify security-relevant requirements can be improved using this approach. We illustrate our approach by providing a step-by-step example of how we improved the security requirements engineering process at the European Telecommunications Standards Institute (ETSI) and report on experiences made in this application.

Keywords: secure software engineering, requirements analysis, organisational learning, requirements workflow modelling

1
2
3 **Noname manuscript No.**
4 (will be inserted by the editor)
5
6
7
8
9

10 **Enhancing Security Requirements Engineering** 11 **by Organisational Learning** 12 13 14 15 16 17 18 19 20 21

22 the date of receipt and acceptance should be inserted later
23
24

25 **Abstract** More and more software projects today are se-
26 curity-related in one way or the other. Requirements engi-
27 neers often fail to recognise indicators for security problems
28 which is a major source of security problems in practice.
29 Identifying security-relevant requirements is labour-inten-
30 sive and error-prone. In order to facilitate the security re-
31 quirements elicitation process, we present an approach sup-
32 porting organisational learning on security requirements by
33 establishing company-wide experience resources, and a so-
34 cio-technical network to benefit from them. The approach
35 is based on modelling the flow of requirements and related
36 experiences. Based on those models, we enable people to ex-
37 change experiences about security-requirements while they
38 write and discuss project requirements. At the same time, the
39 approach enables participating stakeholders to learn while
40 they write requirements. This can increase security aware-
41 ness and facilitate learning on both individual and organi-
42 sational levels. As a basis for our approach, we introduce
43 heuristic assistant tools which support reuse of existing se-
44 curity-related experiences. In particular, they include Baye-
45 sian classifiers which issue a warning automatically when
46 new requirements seem to be security-relevant. Our results
47 indicate that this is feasible, in particular if the classifier is
48 trained with domain specific data and documents from pre-
49 vious projects. We show how the ability to identify security-
50 relevant requirements can be improved using this approach.
51 We illustrate our approach by providing a step-by-step ex-
52 ample of how we improved the security requirements engi-
53 neering process at the European Telecommunications Stan-
54 dards Institute (ETSI) and report on experiences made in this
55 application.
56
57
58

59 **Keywords** secure software engineering, requirements
60 analysis, organisational learning, requirements workflow
61 modelling
62

63 Address(es) of author(s) should be given
64
65

1 Introduction

The growing complexity and interoperability of today's soft-
ware systems creates new security challenges. Even compo-
nents and features that are initially not considered security-
relevant may compromise security when combined with
other features. In a complex software system, requirements
and components are provided by a variety of project part-
ners. In order to close security loopholes, potential problems
should be detected as early as possible during the develop-
ment process. Requirements that may eventually affect sys-
tem security need to be checked carefully before being im-
plemented.

However, identifying those requirements is difficult:
Complex business processes, organisational needs, and crit-
ical assets are handled by software systems. Thus, specifi-
cations from different project partners are voluminous and
contain many requirements. Security requirements may be
implicit, hidden, and spread out over different documents.
Any bug or unforeseen feature interaction in the systems
can increase its vulnerability and diminish system security.
Stakeholders often miss security-related requirements be-
cause of their limited security expertise and experience in
assessing security implications.

Threats to security are a moving target: Attackers find
new security breaches - and security experts develop new
strategies to eliminate them. The body of security expertise
is not static. Knowledge and experience is growing on both
sides. Continuous learning about security requirements and
implied vulnerabilities is indispensable.

From an organisational perspective, *one of the biggest
problems in security engineering* is the lack of experts.

The basic idea of our research is to address this problem
by reducing the dependency on experts by applying experi-
ence-based tools (e.g. HeRA, the Heuristic Requirements
Assistant [26], see Section 4) were possible. This sets free

resources for tasks that cannot be delegated to computer tools. Furthermore, non-experts learn while interacting with experience-based tools due to the feedback they receive. We relate this idea to the existing concepts of organisational learning.

Organisational learning in general comprises the following aspects (cf. [40]):

1. Competent individuals
2. Organisation-wide collection of knowledge and experience, independent of individuals
3. Cultivation of infrastructure for exchange across stakeholders, experts and stored experience

An organisation needs to provide opportunities for learning and incentives for applying what has been learned. As more developers acquire basic or even advanced knowledge in security, the shortage of competent personnel can be mitigated. Our approach can substitute experts, at least for a while. At the same time, it helps stakeholders to develop their security expertise.

Organisational learning faces different challenges in different domains. The specific constraints and challenges determine how the above aspects of organisational learning can be instantiated in the domain of assessing the security-relevance of requirements. As a result, processes, workflows, and tool support are enhanced in an integrated way.

In previous papers [14,23], intermediate *results* of this improvement effort were reported. The dedicated requirements engineering tools we developed include the HeRA heuristic requirements assistant [26] and its extension by Bayesian filters [14].

This paper focuses on the *approach of improving information flows by applying heuristic requirements tools* in the development process. We address the three aspects of organisational learning in the specific case of security requirements (see Table 1). We describe how our approach was applied to the requirements elicitation process of the European Telecommunications Standards Institute (ETSI) in order to enhance their ability to identify security requirements early. ETSI is a major standardization organization within the telco domain and was responsible for the standardization of GSM, UMTS and LTE (2G, 3G and 4G). ETSI is member-driven; members include ISP, smart card providers, network providers, and others and spans across Europe, Asia and the US. One should however note that the approach is independent of the ETSI environment. It can be applied to other environments for taking best advantage of their respective experts, resources, and for tailoring workflows.

In order to demonstrate how the security requirements gained in the elicitation phase can be integrated in the system design phase, and in order to feed back previous experience from designing secure systems into the elicitation phase, we use the security extension UMLsec [18] of the

Unified Modeling Language (UML). This extension allows the system designer to include security requirements and other security-relevant information within the system design models created with the UML notation. There also exist tools which allow the designer to verify the UMLsec models against the security requirements that are included to make sure that the design supports the requirements [17].

The remainder of this paper is organised as follows. Section 2 describes and discusses the challenges of continuous learning in requirements engineering for security-sensitive systems. In Section 3 we show step by step how we integrated learning into the ETSI security requirements process. Heuristic tool assistants are needed to enact that new process and flow of experience. In Section 4 we show how those tools can be used to substitute the presence of a security expert in requirements elicitation. In particular, we present how Bayesian classifiers can be integrated to improve security awareness and provide results of an evaluation (Section 5). In Section 6 we reflect on the presented results and point out future directions. Section 7 outlines related work. Section 8 concludes the paper.

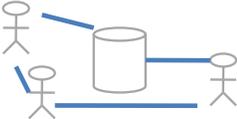
2 Organisational Learning on Security Requirements

Security experts are in high demand in an organisation and have no extra time to spend on documenting their experience. Many of them struggle to keep informed of new security developments while working full time in projects. That makes security experts a scarce resource. Development organisations must use those experts as efficiently as possible. Security experts will need to focus on the most critical and demanding projects and tasks.

As a result, there are often no security experts available to support other projects. Even specific phases during the development of critical systems might not have a security expert assigned. Those projects run a significant risk of overlooking security issues in requirements. Weaknesses found only later during implementation are much more costly to handle. We propose to support security experts by sharing their expertise if no human expert is available. At least part of their experience must be stored or encoded in an appropriate way and brought to bear when requirements are written or discussed. This highlights the necessity for organisational learning. It also points to the specific challenges organisational learning faces in security requirements elicitation and analysis (see Table 1):

1. *Individual learning* is difficult under the strict time pressure of a software project. How can individuals be encouraged and enabled to invest in learning while they work?
2. *Sharing experience and documenting it* is a key challenge: Security experts are already the bottlenecks in an

Table 1 Overview of organisational learning in security requirements engineering. The table summarizes the specific challenges in this environment. The main characteristics of our approach are related to the aspects of organisational learning. The last row points to the concrete example used in this paper in order to illustrate the respective aspects of our approach.

Aspect	Individual learning	Infrastructure for exchange	Collection of expertise
Visualisation			
Challenge in Security RE	Time pressure in projects. Finding time for learning. Motivation to invest time.	Invisible network of workflows&dependencies. Organizing flow of requirements and tacit security experience.	Experts are bottleneck. Must not be distracted. No additional effort can be spent for capturing or documentation.
Approach	Interactive identification of security requirements using tool. Reuse of experience.	Easy-to-use graphical models for analyzing and discussing improvements explicitly.	Reuse existing specifications. Encode experience in heuristic rules etc.
Instance/ Example	Stakeholder use HeRA and Bayesian classifier like an RE-specific spellchecker and learns from feedback.	Step by step Scenario of designing a tailor-made workflow.	Heuristic critiques and training data of Bayesian classifier incorporated in HeRA.

organisation. They cannot carry an additional burden of experience documentation. How can their experience on security be captured and stored without consuming even more of their time and attention?

3. *Infrastructure for exchange* refers to workflow, networks of people and tools. That infrastructure must bring the distributed expertise and experience to bear on a given project. How can the sophisticated flow of requirements and security experience be organized and designed? Much of the experience is tacit [35]. It rarely gets documented.

Individual learning is related to organisational learning as one of its aspects by the above definition. The benefit of *individual learning* (aspect 1) for the organisation is often a one-way relationship: The organisation benefits from the individual learning effort, but there is little gratification in return. Our approach facilitates individual learning as a side-effect of organisational learning aspects. The key to intertwining these two modes of learning lies in an interactive application of heuristics during a stakeholder workshop or direct interaction. Tools and techniques need to be developed and adapted to support that vision. We suggest to co-evolve tools, infrastructure, and flow of requirements and experiences. Stakeholders use the tools and witness the identification of security-related requirements which they might either have overlooked – or not considered a security issue.

Kelloway and Barling [19] point out that each individual knowledge worker needs to have (1) the ability, (2) the motivation, and (3) the opportunity to engage in knowledge work. While ability is a personal property, an organisation

needs to provide motivation for learning and opportunities for applying what has been learned. The infrastructure and tool support we are going to present below is tailored to encourage learning and the application of knowledge. Mostly, stakeholders and knowledge workers should see the immediate advantage of removing security risks early.

When more developers acquire basic or even advanced knowledge about security issues, the shortage of competent personnel can be mitigated. Our approach can substitute experts in some cases and mitigate their absence to some extent in other cases. At the same time, it helps stakeholders to develop their skills in security requirements engineering.

Establishing a *collection of experiences and requirements documents* addresses aspect 2 of organisational learning. Experiences on requirements and how they affect security need to be collected and stored in a reusable format. We address this issue by using the infrastructure of the Heuristic Requirements Assistant (HeRA) [26]. HeRA allows to encode experiences as heuristic critiques based on a script language [27]. In addition, it is possible to capture knowledge about identifying security-relevant requirements with help Bayesian classifiers (as discussed in Section 4 and in [23]). Both mechanisms store experiences in a reusable format, which is important for reducing the strong dependency on experts. When experts are completely or partly replaced by experience-based tools, they gain free time.

For addressing aspect 3 of organisational learning, our goal is to feed back previous experience from designing secure systems into the elicitation phase. To achieve this, we make use of the security extension UMLsec [18] of the Uni-

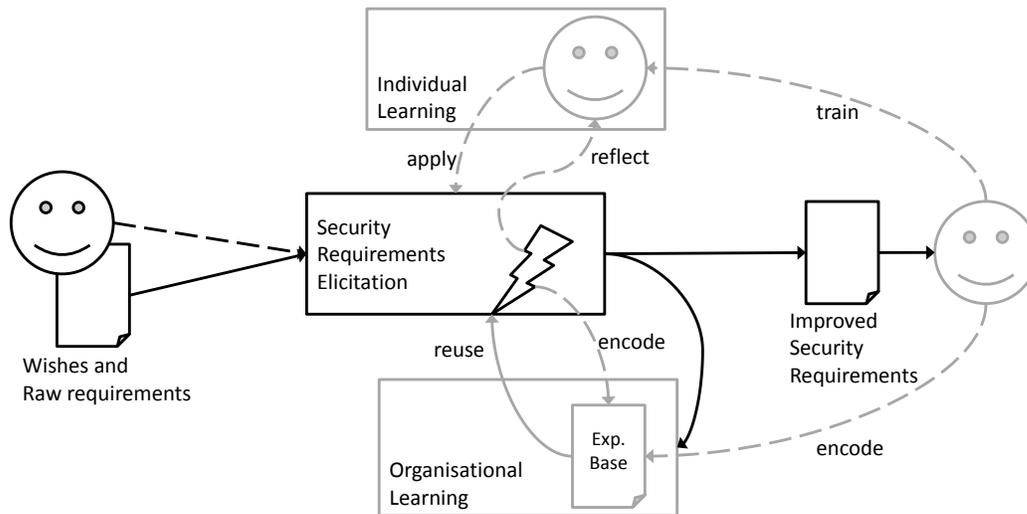


Fig. 1 Learning model, integrating individual and organisational learning.

fied Modeling Language (UML), which allows the designer to document security-relevant information as part of UML design models and is therefore suited to document and transport security design experience.

By reusing these experiences and maybe existing evaluations of requirements, our elicitation tool HeRA assists its users in specifying and analysing requirements with security implications. This *infrastructure for applying experiences* further drives the organisational and individual learning processes: As HeRA benefits from reused experiences, it becomes more effective, fewer problems get passed on, and individuals learn more as they interact with HeRA. Experts do not need to be involved all the time. They gain valuable time for other important security tasks.

It is characteristic of our approach that the documentation of an improved process does not deprive individuals of their important expert role – instead, it helps them to develop their individual experience further.

All three aspects of organisational learning also contribute to individual learning and qualification, when collected and reused experience is fed back into the discussions amongst the stakeholders.

Figure 1 shows our learning model. Learning takes place during the activity “Security Requirements Elicitation”. We differentiate between organisational learning and its individual learning aspect.

1. *Individual learning*: Any participating individual can apply his or her specific experiences. Through reflection, individuals learn while acting, which is a very effective way of learning. This mode of learning is supported by constructive breakdowns that allow individuals reflect in action whether improvements are possible [42].

2. *Organisational learning*: Tools like HeRA can leverage an experience base. They analyse the security requirements created in the activity and compare them to experiences encoded as heuristic critiques. If a critique fires, experiences from the organisation get reused - the experience-based tool shows a message, thus provoking a constructive breakdown. This breakdown triggers reflection (see individual learning above). If a critique from the tools is incorrect, individuals may choose to change the heuristic rules that automatically detect the critiques applicability (called encoding in Figure 1). By this, the experience is added to the experience base.

In our case, it is important to introduce experience and expertise from experts that are not participating in the elicitation task. Thus, we add two more experience flows from the security expert to the individual (training) and to the experience base (encoding of experience as heuristic rules). *Encoding* experiences or *teaching* individuals is still a time-consuming task for the expert, but it will lead to improved security requirements in the long term. Nevertheless, we try to uncover other sources of experience that are cheaper. One of these sources is discussed in depth in this paper: with the help of Bayesian classifiers we train HeRA to automatically identify security relevant requirements. The classifier is trained with requirements classified by the security expert in older versions of the requirements document.

In the following sections, we present the main parts of our approach: Modeling the flow of requirements and experience as the backbone for workflow infrastructure (Section 3). Novel activities are designed into that workflow; they require support by heuristic assistant tools. We present the HeRA tool and its newest extension, Bayesian classifiers (Section 4). Those tools are characteristic of our approach and represent concepts that can be directly reused in other

environments. Evolving workflows, on the other hand, contain both general and environment-specific elements. A new environment will need to update those models accordingly.

3 Improving the Flow of Requirements and Experience

In this section we describe step by step how we improved the situation at ETSI by modeling, analyzing and improving the information [46]. We used the FLOW notation [39] for modeling the sequence of situations and improvements. FLOW was created as an information flow modeling language which covers experience and both documented and un-documented (fluid) information, e.g. requirements [37, 44]. It contains only a few simple symbols and arrows, as it is supposed to be used for discussions [32]. All elements will be introduced below as we need them to design improved flows. In [38], we compared the FLOW notation to a number of related modeling notations, such as data flow [10], process modelling [50], workflow [36], or UML. Those and other notations may be used within our approach. The steps presented below illustrate that it was feasible and useful to use FLOW for that purpose.

3.1 Initial Situation at ETSI

Raw requirements from different sources flow into a given software project. Their quality is rather poor: requirements are inconsistent, ambiguous, and contain not enough detail for security considerations. The process is not yet structured in any way and depends on the few security experts available. This situation is sketched in Figure 2.

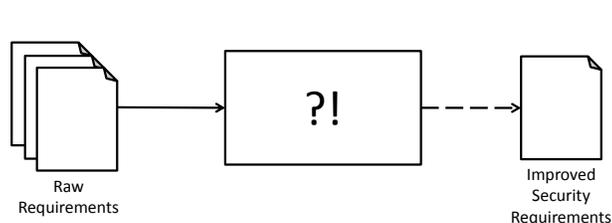


Fig. 2 Initial Situation: Many raw requirements are transformed manually into an improved security requirements specification. We only know that *Somehow*, this is done by security experts.

Model: The *document symbols* represent documented requirements or specifications. Three symbols together represent several documents of the same type. Raw requirements are transformed into improved security requirements during an *activity*. The activity is modeled by a rectangle.

Since we know nothing about it yet, it is labeled with a question mark instead of an activity name. Arrows denote the *flow* of requirements.

Analysis: The unstructured activity does not provide any support or guidance. Only competent security personnel is able to carry out the transformation. The initial situation depends entirely on their capability - and on their availability. As we discussed in the introduction, they are often not available. This situation leads to the above-mentioned security problems.

3.2 First Improvement: Guidance by Common Criteria

For improving this situation, ETSI is using guidance by *Common Criteria* [15]. Common Criteria is a collection of publicly available security domain experiences and guidelines as the standard is for free. However, the language used in the standard require security expertise to understand and make use of. This is what ETSI has provided guidelines for and which we encode as experience in our approach. The flow of requirements is improved by structuring the activity of transforming *raw requirements* into *improved security requirements*. That transformation is broken into a series of refinement steps, as shown in Figure 3. This process guides the stakeholders to first think about security needs on an abstract level, such as the need of identification of users to an application. It then guides the stakeholder to refine these abstract statements into SMART security requirements through a number of steps, as shown in Figure 3. The refinement can be looked upon as a number of questions arising from the structure of ISO 15408 helping the stakeholder both in the refinement and in the specification of the security requirement.

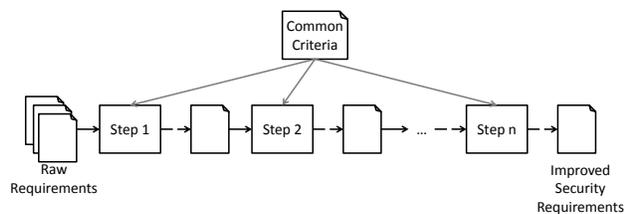


Fig. 3 Situation 2: Common Criteria guides the refinement of raw requirements into improved security requirements. A sequence of refinement steps provides experience from the Common Criteria for each step (gray arrows).

Model: In addition to the flow of requirements (black arrows), the availability of experience on security requirements elicitation is crucial for the successful execution of the refinement tasks. The *gray arrows* from Common Criteria represent that flow of experience. Experience controls how the activity is being carried out. For example, the exact

way of refining a requirement is being controlled by experience. The requirement itself is considered the input this activity works on. In the model, input flow (e.g., requirements) is attached to the activity rectangle from the side. Experience is attached to the top of the rectangle, which indicates that it is not considered input, but control. At his point, only experience documented in Common Criteria has been considered.

Analysis: This structure provides guidance, but cannot compensate for missing expertise and experience. The Common Criteria document provides a static structure. It does not dynamically adapt to new findings or known problems at ETSI. Defining refinement steps offers a break-down structure for requirement analysis. The wording of Common Criteria is directed towards security personnel. It is difficult to understand and apply in practice. This situation constitutes a formal guidance rather than content-oriented support. Hence, the lack of security experts remains a problem.

3.3 Insight: Considering People

Since security experts are the main bearers of experience, this fact should be modeled and optimised explicitly (Figure 4). Many other project participants lack awareness of security and the importance of identifying respective requirements early. Therefore, we explicitly include people into the model.

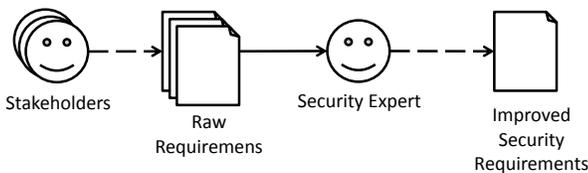


Fig. 4 Stakeholders write a document with initial raw requirements each

Model: Requirements documents are written independently. There are several stakeholders (e.g. representatives of customers or partners) and their documents, depicted by three overlaid symbols. All stakeholders are on their own. The flow or requirements from those stakeholders to their respective documents has particular properties: It can hardly be repeated literally, since people forget what they wrote. The flow can be disrupted easily as they are disturbed during writing. In a metaphoric expression we call this fluid information. It is quick and easy to transfer but it may be spilled and lost. This is an important difference to so-called solid information contained in a document.

Analysis: We consider it essential to model documented (solid) and non-documented (fluid) information in our approach. Both types exist side by side and need to be taken seriously. The intention of our models is to create a balanced and appropriate network of forward and backward flows, both solid and fluid. So far, the new model represents an insight rather than an improvement. Resulting specifications still need to be integrated by an expert. This situation also does not accommodate learning. Stakeholders tend to repeat their same problems over and over again, since they receive no feedback on their specifications. However, this insight stimulated searching for alternative flows during elicitation.

3.4 Improvement: Encouraging Direct Communication in Workshop

Isolated stakeholders could not help each other, or benefit by learning. Inspired by organisational learning aspect 3 (infrastructure), we considered interactive workshops for elicitation. When stakeholders write their requirements in such a workshop (see Figure 5), they need less preparatory effort, and they interact heavily. However, an experienced person will be needed to summarize the discussion and write requirements. A requirements engineer might be appropriate for that task.

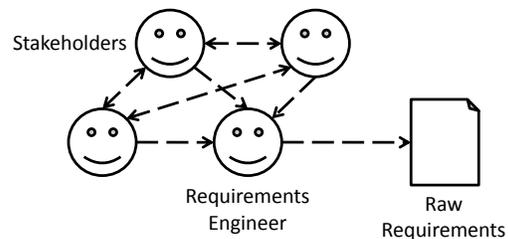


Fig. 5 Requirements are specified in a workshop.

Model: There are no new FLOW symbols in this model. The extensive use of dashed lines indicates the fluid nature of direct communication in a workshop. Depicting stakeholders separately allowed us to highlight the communication between them – which did not occur with isolated stakeholders.

Analysis: Talking is often considered faster and more convenient. When one stakeholder raises requirements, others may react to them or even identify security risks. An experienced requirements engineer who is not a security expert, however, will be mainly concerned with requirements elicitation and capturing.

3.5 Improvement: Elicitation Pattern with Tool

An important improvement was the introduction of tool assistance. HeRA is the Heuristic Requirements Assistant tool. It uses heuristic rules for scanning requirements and issues warnings when it detects potential problems. We applied this principle on a wide range of heuristics [26]. In the context of the security identification task, HeRA was equipped with heuristics to identify security-relevant requirements.

We designed an elicitation pattern as it occurs in requirements engineering: HeRA helps by partially replacing security experts. If at all, the expert provides guidance during the writing of requirements. The expert may be substituted by a stakeholder typing or copying into HeRA. Figure 6 depicts this scenario.

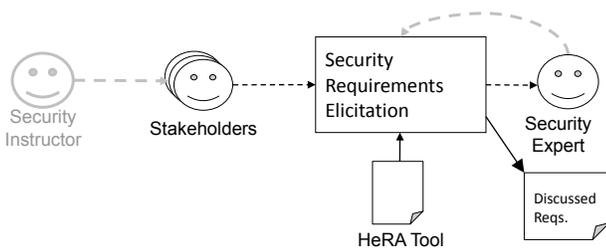


Fig. 6 Requirements Elicitation supported by a tool, with typical flows.

Model: The distribution of fluid vs. solid arrows characterizes the type of communication that determines this workflow. In this case, it is a careful mix of documented (solid) and fluid flows. Security instructors provide basic security knowledge and awareness. Their gray arrow stands for the experience they transfer to stakeholders. Along the same lines, the security expert mostly helps by controlling the elicitation activity. Again, this is experience (gray) in security handling, not content knowledge of specific requirements (black). HeRA is depicted as a solid part. It is connected to the activity rectangle from below. Like in SADT, this indicates that HeRA supports the activity.

Analysis: HeRA checks stakeholder requirements by its heuristic rules. Whenever it issues a warning, the potential problem is discussed and the expert can facilitate that discussion. If there is no warning, requirements from a stakeholder are accepted faster. This first check speeds up security consideration and helps to save expert time. The pattern consists of flows of experience from security-aware personnel. Both the instructor and the expert are not domain experts and should focus on security aspects. This pattern allows them to do that. The HeRA tools acts as an interactive editor for requirements; it also produces the solid output: a set of requirements that have been checked for security implications. From the perspective of organisational learning,

HeRA rules represent encoded experience. The focused discussions following a warning reach two goals at a time: a specific security issue in a project is resolved; and all participating stakeholders receive an intense lesson in security as a side-effect. This addresses the challenges of organisational and individual learning: avoiding additional effort. Note that many of the flows in this pattern are fast and fluid but solid HeRA stimulates them.

3.6 Improvement: Experience Reuse from Previous Projects

Organisational learning calls for an infrastructure for exchange. The above models show local patterns of flow. The elicitation pattern is the result of careful consideration on the level of requirements and experience flows. The use of HeRA represents the application of collected security experience in the form of heuristic rules. To get beyond the local improvements, we sketched and designed an infrastructure of flows that would combine several of the above models – and reach out for reuse of requirements from previous projects 7.

Model: Figure 7 describes the complete process and flow of security requirements elicitation. Although this Figure is rather complex, it contains no new FLOW elements.

Analysis: There are three parts of this model: The elicitation pattern on the right side feeds into the activity in the center. That activity refers to the above-mentioned five-step refinement strategy at ETSI (see [23]). It receives the elicited and discussed requirements with marks for potential security problems. It is controlled by solid security experience that was derived from Common Criteria and other sources (gray, from top). There is also a reuse loop of specific experiences and insights. It represents the case when a participating individual feeds back observations made in a project. On the right side, there is the system construction part. It relies on the UMLsec tool. In this phase, design decisions must be taken. This is the point at which designers must consider security. They may gain new insights in the requirements that caused security considerations in design. This is the time to encode this insight into HeRA rules – for the benefit of all future projects. They will be warned as early as during the elicitation activity.

3.7 Latest Improvement: Solid Feedback

The feedback from UMLsec to HeRA rules in Figure 7 is fluid: Some expert or participant of UMLsec system construction is required to take time to encode the insight into a heuristic rule. Organisational learning challenges remind us that this requires altruism and might not always happen

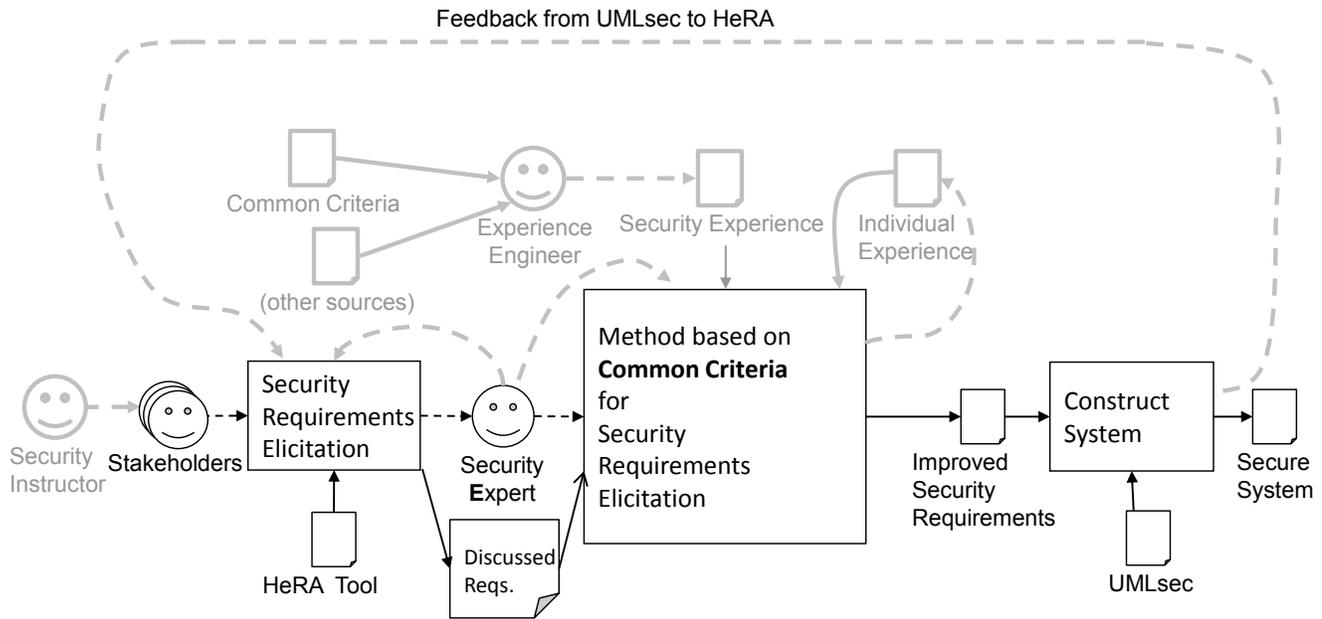


Fig. 7 The overarching flow infrastructure of our so-called SecReq model: HeRA supports the security requirements elicitation by offering rule-based critique. Downstream activities from construction feed back to inform security requirements elicitation.

under time pressure. We, therefore, envisioned a solid feedback flow that would cause no or very limited overhead.

In terms of the FLOW model, we wanted to add a solid flow from the end of the central refinement activity to elicitation. After the five-step refinement process, several security implications have been found. The key is to make those insights available to future projects – and at a much earlier point in time: during elicitation. Although the change in the model is simple, its implementation is not. Improving the flow infrastructure requires new tools to support it. In this case, we found Bayesian classifiers as a concept for extending HeRA. While the overall model remains almost the same (see Figure 7), a closer look needed to be taken at the elicitation activity, as shown in Figure 8

Model: Some elements have labels attached. They represent comments and are supposed to clarify the interfaces between this refined Figure and the overall model (see Figure 9 in Section 4).

Analysis: Bayesian classifiers are explained in Section 4. At this point, flow modeling is useless without implementing the tool features to support it. This model highlights that Security Requirements Elicitation will now be controlled by Bayesian classifiers in addition to the other heuristic rules used in HeRA. Bayesian classifiers need training sets before they can evaluate a new requirements document for security relevance. Our vision was to use requirements that had gone through the five-step refinement process for training.

Both feedback loops enhance organisational learning:

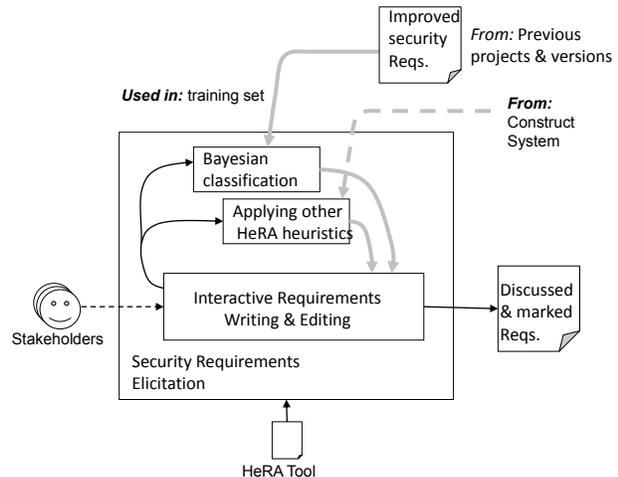


Fig. 8 Details of the activity *Security Requirements Elicitation* in Figure 7. Bayesian classifiers are envisioned to reuse feedback from requirements of previous projects

1. Explicit knowledge from designing secure systems is encoded in HeRA's heuristics.
2. Classification knowledge automatically captured by HeRA's Bayesian classifier.

From an organisational learning perspective, infrastructure and tool support are enhanced in an integrated way. Individual stakeholders benefit from all kinds of feedback: they all contribute to HeRA's ability for security warnings. As pointed out above, those warnings trigger discussions that help stakeholders to learn. Tools like HeRA and doc-

uments like the improved requirements from the refinement process represent collections of experience. Models and their implementation provide infrastructure for exchange. The three components of organisational learning have been applied to security requirements engineering.

4 Heuristic Assistant Tools for the Experience Reuse

Our SecReq approach assists in security requirements elicitation. It provides mechanisms to trace security requirements from high-level security statements, such as security goals and objectives, to secure design [14]. We aim at making security best practices and experiences available to developers and designers with no or limited experience with security. SecReq integrates three distinctive techniques (see Figure 9): (1) Common Criteria and its underlying security requirements elicitation and refinement process [15], (2) the HeRA tool with its security-related heuristic rules [26], and (3) the UMLsec approach for security analysis and design [18].

During Section 3 the model in Figure 9 was developed step by step. In this Section the working of Bayesian classifiers in the HeRA tool will be introduced in order to implement that vision.

4.1 The HeRA Requirements Assistant

As pointed out, there are not many security experts, and most security guidelines or "best practices" are written by and for security experts. Also, security best practices such as standards ISO 14508 (Common Criteria), ISO 17799 are static documents that do not account for new and emerging security threats. Security issues can be characterized as known or hidden, generic or domain-specific. Normally, a security expert is absolutely necessary to identify *hidden* security issues, while *known* issues can be identified using security best practices.

SecReq - and the HeRA tool in particular - guide the translation of these best practices into heuristic rules. They try to make better use of the few security experts around. Rather than having experts do the identification and refinement of all security issues, SecReq reuses their expertise and makes their security knowledge available to non-security experts.

The basic idea behind HeRA is the use of heuristic tools to analyse natural language requirements documentation and to offer useful feedback [26,22]. Feedback is either constructive critique or a derived model. Both types of feedback proved to be valuable to increase quality and productivity when documenting requirements.

Derived models. Heuristic tools are able to automatically derive models from textual use case descriptions [26].

UML use case diagrams *give immediate feedback about the context* of the textual use case that is currently edited. Graphical process models show how a set of use cases interacts to support a global business process [24]. Automatically derived use case point models help to *identify problems* like *requirements creep*, i.e. unperceived growth of demanded functionality over time [26].

Automatically derived models *offer analysts additional information* that helps to *make good decisions* when documenting requirements. In addition, this kind of feedback allows analysts to regard their requirements from a different point of view. Evaluation showed that this helps to *assess the consistency and completeness* of a given requirements document.

Heuristic critiques. Heuristic critiques consist of a heuristic rule, a meaningful message, and a criticality. The heuristic rule can be used to analyse natural language requirements and to identify situations where the critique is appropriate. In this case the meaningful message is displayed as a warning, an error, or a hint, depending on its criticality.

Heuristic critiques can be seen as an experience package with a strong focus on reuse [27]. Evaluation showed that *users write better requirements* documents with this kind of experience support [25]. Evaluation showed that typical users are able to adjust existing rules or to create new heuristic critiques [22].

4.2 A Bayesian Filter Extension for HeRA

In this paper we describe an improved version of HeRA. We intended to reduce the amount of manual work by making better use of documents and experience from previous projects. Figure 9 shows this improvement as a solid arrow from improved security requirements from previous projects back to the early security requirements elicitation activity (shaded box). We use them to continually train a Bayesian filter. Growing numbers of pre-classified requirements from previous projects will increase the classification ability of that filter. This new experience flow improves the effectiveness and efficiency of the SecReq approach, because it reduces manual work by leveraging Bayesian classifiers. This enables HeRA to address both generic and domain-specific security aspects and to capture experts' tacit knowledge better. Based on this knowledge, heuristic computer-based feedback can simulate the presence of a security expert during security requirements elicitation.

4.2.1 Classifying Security Requirements.

For training the Bayesian classifier, we need pre-classified requirements. During expert classification, we encountered three different types of security requirements. We define:

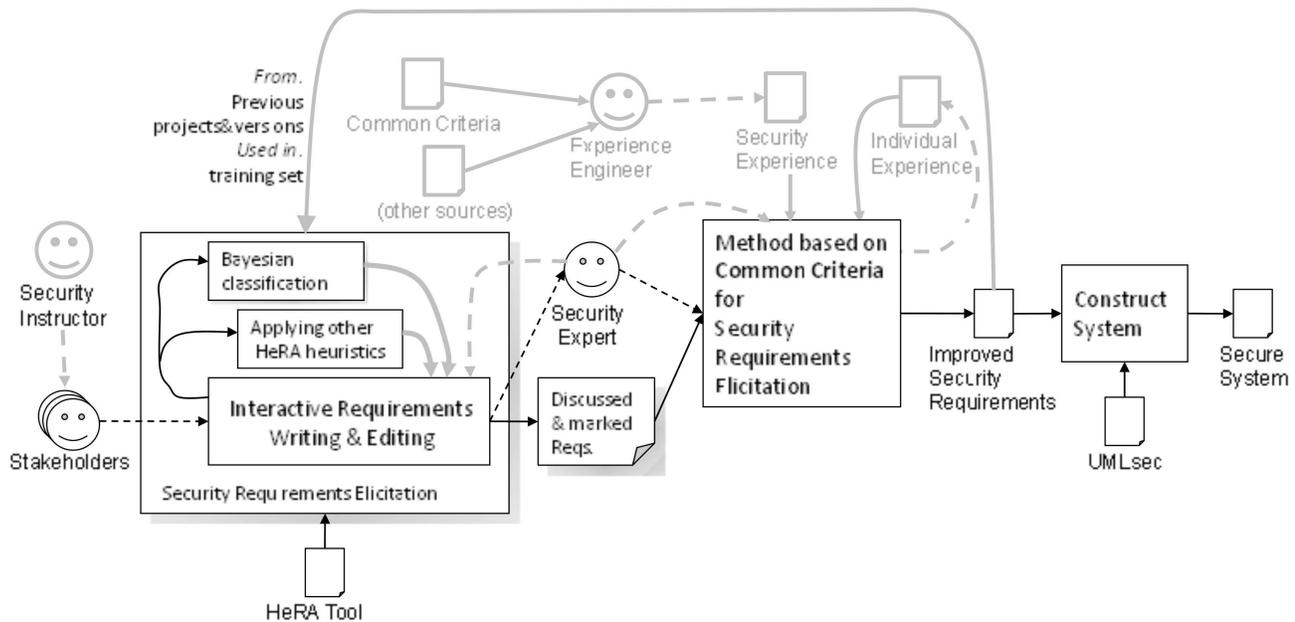


Fig. 9 Overview of SecReq approach with solid feedback from UMLsec to HeRA. In this model, Security Requirements Elicitation is supported by applying Bayesian classifiers. They are trained by reusing documents from earlier projects. Specifications from previous projects are used to train the classifiers. This establishes a solid feedback.

Security requirement:

- (i) A (quality) requirement describing that a part of the system shall be secure, or
- (ii) a property which, if violated, may threaten the security of a system.

In our context, security requirements are the result of refining security-relevant aspects. Our goal is to support this process.

Security-relevant requirement:

- (i) A requirement that should be refined into one or more security requirement(s), or
- (ii) a property that is potentially important for assessing the security of the system.

Example: “The card must ensure that the transaction is performed by the same POS device as was used for the purchase being canceled [...]”

During pre-classification, we encountered another type of requirements:

Security-related requirement:

- (i) A requirement that gives (functional) details of security requirements, or
- (ii) a requirement which arises in the context of security considerations.

Example: “The card and the PSAM must use a public key algorithm for mutual authentication and session key exchange [...]”

To support the identification of hidden security aspects, we need to identify *security-relevant requirements*. It took our experts some training to avoid false classification (e.g. classifying a security or security-related requirement as being security-relevant). Furthermore, each and every functional requirement could be regarded to be *somewhat* security-relevant: Safety and Confidentiality of data should always be ensured. Hence, we need a good classification strategy for manual classification. The *classification question* was very instrumental when classifying a requirement:

Classification Question:

Are you willing to spend money to ensure that the system is secure with respect to this requirement? Assume there is only a limited budget for refining requirements to security requirements and that there is a need to prioritize and balance cost and risk.

Outputs from security risk analysis approaches (such as CORAS [11], CRAMM [4], and OCTAVE [1]) can be used to support such evaluations, as these provide lists of threats, their related risk level, and potential consequences. Some approaches also directly consider potential monetary losses. The question then becomes:

Can you afford to not invest to reduce or remove relevant risks?

5 Evaluation

Our approach of applying organisational learning to requirements engineering in secure system development has two parts: modeling and design of flows, and the implementation of dedicated tools to support those flows. Section 3 was devoted to a detailed step-by-step presentation of an evolving flow model. We used the FLOW notation [38] which was designed for modeling flows of requirements and experiences. However, other notations might also be used for that purpose. The sequence of models [38], analyses [46], and conclusions [44] demonstrates the reflective process involved with improving infrastructures [45]. Solid versus fluid information and the distinction between requirements and experience flows were among the key concepts of modeling.

That discussion was intended not only to motivate the final result of Figure 9. It also provided one case to demonstrate the feasibility of modeling variants and improvements with a simple notation. Other environments than ETSI will need to consider their particular stakeholders, conventions, and prescribed flows when they apply our approach.

In the remainder of this Section, the latest extension, Bayesian Classifiers will be evaluated in depth.

5.1 Evaluation of Bayesian Classifiers

This section discusses the quality of classifiers and how they can be used to assist in security requirements elicitation. First, we define our evaluation goals in Section 5.1.1. Then we describe our strategy to reach these goals and the general process of evaluation in Section 5.1.2. Finally, we show and discuss the results for each evaluation goal in Sections 5.1.3, 5.1.4, and 5.1.5.

5.1.1 Evaluation Goals

In order to evaluate our Bayesian classifiers, we define three evaluation goals:

- (G1) Evaluate accuracy of classifiers for security-relevant requirements.
- (G2) Evaluate if trained filters can be transferred to other domains.
- (G3) Evaluate how useful practitioners consider automatically identifying security requirements.

For the goals (G1) and (G2) we used expert evaluation to create meaningful test data, as described in Section 4.2.1. In addition, we derived data from analysing existing requirements databases. Subsets of this test data were used to train and evaluate the Bayesian classifiers. Our evaluation strategy had to ensure that training and evaluation sets were kept disjoint. In the context of this paper, goal (G3) is informally

evaluated by asking experts for their opinions about classification results. See Section 6.3 for the implications of our results on industrial practice. A more formal evaluation remains future work.

5.1.2 Evaluation Strategy

Assessing the quality of machine learning algorithms is not trivial:

- *Use disjoint training and evaluation data.* We must not use the same requirements for training and evaluation.
- *Select training data systematically.* For reproducible and representative results, we need to systematically choose the requirements we use for training.
- *Avoid overfitting.* We need to show that our approach is not limited to the specific test data used. Overfitting happens when the Bayesian classifier adjusts to the specific training data.

Typically, *k-fold cross validation* is used to deal with these concerns [7, 16]. This validation method ensures that statistics are not biased for a small set of data [48]. The dataset is randomly sorted and then split into k parts of equal size. $k - 1$ of the parts are concatenated and used for training. The trained classifier is then run on the remaining part for evaluation. This procedure is carried out iteratively with a different part being held back for classification each time. The classification performances averaged over all k parts characterizes the classifier. According to [7], we used $k = 10$: With larger k , the parts would be too small and might not even contain a single security-relevant requirement.

We used standard metrics from information retrieval to measure the performance of Bayesian classifiers: precision, recall, and f-measure [3]. Based on the data reported in [16], we consider f-measures over 0.7 to be good. For our purpose, high recall is considered more important than high precision. A classifier is regarded useful in our SecReq approach if precision is at least 0.6, and recall is at least 0.7.

In our evaluation, we used three industrial requirements documents:

- The Common Electronic Purse Specification (ePurse) [6]
- The Customer Premises Network specification (CPN) [47]
- The Global Platform Specification (GP) [13]

As described in detail below, we experimented with various different training sets applied to each of the three real-world specifications.

Table 2 provides an overview of the three specifications we used for evaluation of our classifiers: For each specification (left column), we list the total number of requirements they contain (2nd. column) and the number of requirements considered security-relevant (3rd. column). We used either

Table 2 Industrial requirements specifications used for evaluation.

<i>Document</i>	<i>total reqs.</i>	<i>security-relevant reqs.</i>	<i>security-relevance determined by</i>
Common Electronic Purse (ePurse)	124	83	expert
Customer Premises Network (CPN)	210	41	database
Global Platform Spec. (GP)	176	63	expert

experts (see Sect. 4.2.1) or existing databases for identifying security-relevant requirements (last column).

5.1.3 Accuracy of Security Classifiers: G1

To test the accuracy of the Bayesian classifier, we use 10-fold cross validation on each of our classified specifications. In Figure 10 we also show the results for smaller training sets. *Training size* gives the number of parts in the 10-fold cross validation considered for training. The trend shown in Figure 10 helps to evaluate whether the training set is sufficient. Results exceed the above-mentioned thresholds for recall and precision. Hence, we consider the classifier useful.

5.1.4 Transferability of Classifiers Trained in a Single Domain: G2.a

Classifying industrial specifications manually was time-consuming. It was needed for training the classifiers. Reuse of trained classifiers could reduce that effort. Therefore, we evaluated the quality of classification when we applied a trained classifier to specifications from different projects - without additional training. In order to produce comparative results, we used 10-fold cross validation in all cases, but varied the specifications used for training and for applying the classifiers.

Table 3 shows our results. The first column indicates which specification was used for training. We list the quality criteria (recall, precision, and f-measure) when applying the respective classifier to each of the three industrial specifications in the last three columns. Values on the main diagonal are set in italics: they represent the special case of (G1) reported above, where the *same* specification was used for training and for testing. Even in those cases, the 10-fold cross validation ensured that we never used the same requirements for training and evaluation.

The results in Table 3 are surprisingly clear: f-measures on the diagonal are 0.86 and higher (same specification for training and test). All other f-measures are far below 0.7: whenever we used different specifications for training and evaluation, transferability is very limited. A filter cannot easily be used in a different context.

Table 3 Training classifier with one specification, applying it to another.

Training	Applying to:	ePurse	CPN	GP
ePurse	recall	0.93	0.54	0.85
	precis	0.83	0.23	0.43
	f-measure	0.88	0.33	0.57
CPN	recall	0.33	0.95	0.19
	precis	0.99	0.98	0.29
	f-measure	0.47	0.96	0.23
GP	recall	0.48	0.65	0.92
	precis	0.72	0.29	0.81
	f-measure	0.58	0.4	0.86

5.1.5 Transferability of Classifiers Trained in Multiple Domains: G2.b

If we apply a Bayesian classifier trained with a specification from one domain to a different domain, we get poor results (see G2.a). This could either point to the fact that we cannot transfer classifiers to other domains or that we used a bad training set. To investigate this, we carried out a third evaluation run where the classifier was trained with values from a mix of specifications. For this, we join the requirements from two or three specifications as input for the 10-fold cross validation. The results in Table 4 show: When we used more than one specification for training, the filter became more generally applicable. If we used two specifications in training, the evaluation for the third specification delivered better results than after a single-specification training (G2.a).

Combination of different specifications in training made the classifier more generally applicable. Obviously, classification quality is not only based on domain-specific terms - which would not occur in the second training specification. Thus, a good domain-independent classifier can be created with a sufficiently large training set.

The bottom entry in Table 4 shows the results when we combined all three specifications for training. Now we got good results for all three specifications included in the evaluation. Figure 11 shows the learning curve, by giving the results when using less than 9 parts for training. The learning curve grows not as fast as in the Figure 10, probably because the classifier cannot leverage the domain specific concepts. Nevertheless, we get a recall of 91 %, a precision of 79 %, and a f-measure of 84 % - results that clearly show that the trained classifier is suitable to support security requirements elicitation in all of the three domains used for training.

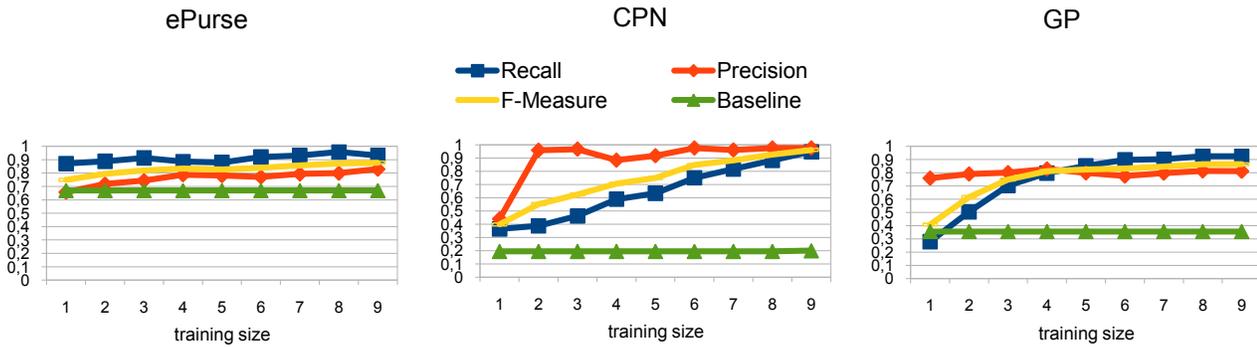


Fig. 10 Results of 10-fold cross validation using only one specification. Baseline is the precision we get when classifying all req. to be security-relevant.

Table 4 Training with more than one specification.

Training	Applied to:	cross-eval	ePurse	CPN	GP
ePurse + CPN	recall	0.93	0.95	0.85	0.56
	precis	0.81	0.80	1	0.51
	f-m.	0.87	0.87	0.92	0.53
ePurse + GP	recall	0.96	0.98	0.85	0.85
	precis	0.80	0.78	0.26	0.8
	f-m.	0.87	0.87	0.40	0.82
CPN + GP	recall	0.87	0.31	0.75	0.88
	precis	0.82	0.84	0.88	0.81
	f-m.	0.85	0.46	0.81	0.84
ePurse + CPN + GP	recall	0.91	0.95	0.85	0.88
	precis	0.79	0.80	0.94	0.78
	f-m.	0.84	0.87	0.89	0.83

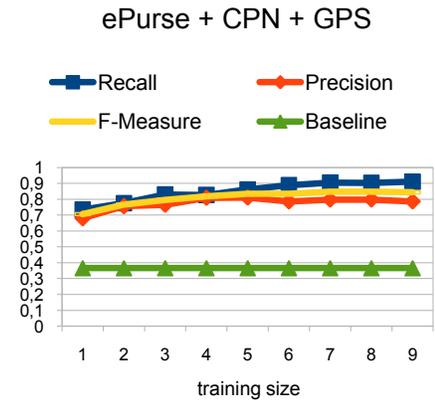


Fig. 11 10-fold cross validation, multiple training

6 Discussion and Implications on Industrial Practice

In Section 5 we evaluated if it is possible to identify security-relevant requirements with help of a Bayesian classifier. It is important to note that for evaluation purposes we did not use the Bayesian classifier in the way it was designed for (compare Section 3):

- *For evaluation* we used a complete specification. Parts of the specifications were used for training, other parts were used for evaluation of recall and precision.
- *In practice* we suggest to use the Bayesian Classifier in an Elicitation tool. Each requirement is classified immediately after it has been written down.

This feedback can be used during an elicitation meeting for immediate clarification on how to proceed with security-relevant requirements. Later, it could be used to generate a list of security-relevant requirements to discuss with security experts. In our SecReq approach, we trigger a refine-

ment wizard that allows laymen to start with the refinement themselves.

In this section we discuss whether the observed results are sufficient for employing the filter in practice at its current status. Then we take a look at the validity of our evaluation of the Bayesian classifier filter. Finally, we summarise the discussion with practitioners and describe how they perceive the implications of the filter in practice, meaning their development projects.

6.1 Interpretation of Evaluation Results

As shown in Section 5 we achieved very good results in cases where the classifier is applied to the requirements from the same source as it was trained with. We also observed poor results in cases where the classifier was applied to a different requirements specification than the one it was trained with. We also observed that the combination of training sets from different sources produces a classifier that works well

with requirements from all sources. This shows that a general classifier for security relevance can be created by applying larger training sets and specifications from more domains.

To summarise, in its current status the classifier is indeed a very valuable addition for example in the context of software evolution or product lines. I.e., the classifier could be trained using the last version of the requirements specification and then offer precious help in developing the new software version. Typically, subsequent specifications resemble their predecessor in large parts and add only small new parts. Evaluation of this situation is covered by k-fold cross validation, as large ($k - 1$) parts of a specification are used for training and applied to a small held-out part. Therefore, the results in Figure 10 apply to this situation. In other situations, the learning curve in Figure 11 and tests with systematic training with falsely classified requirements show that the classifier quickly adapts to new domains.

6.2 Discussion on Validity

Wohlin et al. define types of threats to validity for empirical studies [51]. We consider threats to construct, internal, external, and conclusion validity to be relevant to our evaluation.

Construct Validity. Construct validity deals with the way the evaluation was set up and executed, i.e., the goodness of the evaluation process, the evaluation goals and the distribution of evaluation variables (indirect and direct variables).

In our case, the assumptions made on the classification question and our interpretation of what comprises a good result is critical to determining the goodness of the evaluation. When it comes to the classification question there are many alternative ways to define security-relevance. However, our classification was an effective choice in practice as it helped us to adjust our classification in a way that our security experts could agree on the majority of requirements. Next it is important to consider whether it was sound to apply the classifier on final versions of requirements during the evaluation. This depends on the level of abstraction on which the functional information is presented. In practice the requirements are regularly refined from high level functional requirements to low-level descriptions of security-related aspects.

Internal Validity. Internal validity examines the confidence in the accuracy of the results for the evaluation context.

Concerning accuracy of the results, it is important to assess the way that we handled training of the classifiers during evaluation. We used k-fold cross validation and avoided using identical requirements in training and evaluation, as well as overfitting.

Randomly choosing requirements for training is not the best way to produce a good filter. Ideally, we would train the filter systematically with false positives and false negatives, until it produces good results. Preliminary tests show that this even increases the performance of the classifier with very small training sets.

External Validity. External validity addresses the level of generalisability of the results observed.

In our evaluation, we used three real-world requirement specifications from different domains and authors. We have no reason to doubt the applicability of our approach on different specifications.

Conclusion Validity. Conclusion validity addresses the question, whether the results could be reproduced by others.

We used specifications from two different domains in our evaluation. Therefore, we cannot guarantee that our results would hold for a third domain. To leverage this threat, we invite others to replicate our experiment, or use our results and share our evaluation tool, classified data sets, and the databases of learned words at:

<http://www.se.uni-hannover.de/en/re/secreq>.

6.3 Implications on Industrial Practice (G3)

In practice, there will rarely be budget to tackle all relevant security aspects. Some of them may even conflict. Hence, developers need to get the *right security* (i.e. the relevant and adequate security requirements). For this reason, the classification question (see Section 4.2.1) focuses on money, where money covers both development costs but also the cost associated with the lack of a critical security feature in the end-product (this includes costs, schedule, effort, resources, etc.). When it comes to techniques and tools for security elicitation support, such a tool needs to help a developer getting *security right* (i.e. to implement the security requirements correctly), including being able to separate out the important and prioritised security aspects and hidden security requirements that are somehow concerned with potential business and money consequences (loss and gain). Furthermore, such support must be integrated in a natural way such that the tool supports the way the developer work in the security requirements elicitation process and not the other way around. In practice, spending money on something that is not going to end up in the final system is often considered a waste of time and effort.

The Bayesian classification as an addition to SecReq not only contributes to a more effective and focused security elicitation process, but also in separating important from not so important security-relevant aspects. The Bayesian classification and security expert simulation in HeRA directly enables effective reuse of earlier experience, as well as prioritising and company specific security-related focus areas

or policies. In particular, HeRA provides the ability to train the classification to be system and project specific. The ability to first train the classification engine to understand how to separate important security-relevant aspects from not so important, and then use this newly gained knowledge to traverse functional descriptions and already specified security requirements have a promising potential to contribute in a better control of security spending in development projects.

6.4 Outlook: Training by simulation

Our experience shows that the individual learning aspect is very important. The impact of participating in discussions supported by heuristic tools on security issues is very strong. We envision to use this effect for training. Figure 12 shows this situation in FLOW. Based on a list of raw requirements, security requirements are interactively written. HeRA analyses these inputs based on the various feedback facilities. The stakeholder learns during this simulated security elicitation session by reflecting on the feedback.

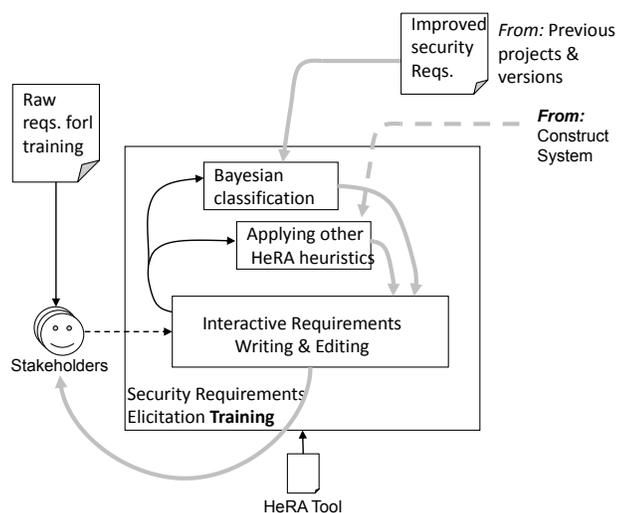


Fig. 12 A stakeholder trains security requirements elicitation.

Advantages:

- No expert needed for training, no instruction needed
- Up-to date experience can be used (same as for productive work)
- Stakeholders and new security people can use training by simulation to get adjusted to the particularities of the environment.
- Repetition of training is simple due to HeRA automation. Improvement can be seen when using the same input.

- Smooth transition from training to learning, even intertwining phases of work with phases of individual learning in the simulator.

7 Related Work

The section focuses on several existing works that are related with our work. We first discuss the state of the art of security requirement engineering process and tool support, then related work about information flow modeling and heuristics and finally the works relating to natural language processing in the requirement engineering domain.

7.1 Security Requirements

A significant amount of work has been carried out on security requirements engineering, in particular relating to tool-support for security requirement engineering. Chung considered a process-oriented approach to develop secure information system [8]. Security goals are considered as a class of criteria for selecting among design decisions and as a part of the overall process including decomposition, satisficing and argumentation methods. A prototype development tool is presented for the security requirements elicitation. The tool includes a graphical interface to view the goal graph expansion process and to interactively browse, select and apply methods, and a textual interface to enter arguments towards a design rationale. Mouratidis, Giorgini et al. propose an argumentation based extension of the i*/Tropos requirements engineering framework to deal with security requirements [33, 12]. The approach allows one to capture high-level security requirements before analysing the specific solution design. Giorgini et al. further introduced the *ST-Tool* for design and verification of functional and security requirements based on the Secure Tropos methodology [52]. The tool supports analysing goals, actors, services, and data of corresponding objects through a GUI interface. The models can be analyzed as to whether they satisfy some general desirable security properties. *SecTro* is an automated modelling tool that also provides support for the Secure Tropos methodology for the development of secure information systems [53]. The tool analyses the security goals, security constraints, task, and resources through a security enhanced actor model. *SecTro* also allows one to analyse the attackers goals and attacks through security attack scenario. Ouedraogo et al. presents an agent-based system to support assurance of security requirements [34]. Based on Secure Tropos, the approach complement security requirements engineering methodologies by gathering continuous evidence to check whether security requirements have been correctly implemented. Matulevicius et al. presents an approach which adapts Secure Tropos for security risk management in the

early phases of information systems development [30]. It allows for checking Secure Tropos concepts and terminology against those of current risk management standards. Sindre and Opdahl propose an approach to eliciting security requirements based on use cases, which extends traditional use cases to also cover misuse [43]. Mellado et al. present the *SREPPLine tool* which provides automated support for the security requirement engineering process for software product lines (SREPPLine) [31]. The tool mainly supports the automation of security requirements management activities involved in SREPPLine. The tool prioritizes the security requirements and generates a security requirement specification document. However, activities that introduce new requirements (i.e. updates of the security feature repository) are performed manually. The *UMLsec* tool [17] supports the analysis of the security aspects expressed in the security extension UMLsec [18] of the Unified Modeling Language (UML). The tool mainly focuses on the verification of the most important security requirements, which can be directly used in the model, together with their formal definitions.

In summary, most of the related work dealing with the management of security requirements has the goal to analyse and verify the requirements through goals, tasks, resources, and design models within the system environment. Furthermore, security expertise is required to operate these tools. In contrast, our work focuses on an approach supporting organisational learning on security requirements by establishing company-wide experience resources, and a socio-technical network to benefit from them. The approach is based on modelling the flow of requirements and related experiences. It can be used in conjunction with the security requirements analysis approaches mentioned above.

7.2 Information Flow Modelling

Winkler uses information flow models to increase traceability in software projects [49]. Damian et al. consider social networks to describe communication in software projects by differentiating media from transfer information, and identifies patterns like "bottleneck" [9]. Schneider et al. propose a simple graphical notation for describing the flow (path) of information such as requirements and security requirements are a special case of the information [38]. This work distinguishes between so-called "solid" (document-based) and "fluid" (e.g. spoken, email, informal) representations. Dashed lines and faces denote flow and storage of fluid information, whereas solid lines and document symbols represent solid information representation. Unlike the work of Winkler in [49], fluid information is modelled explicitly. Dashed lines and faces denote flow and storage of fluid information, whereas solid lines and document symbols represent solid information representation. An interesting observations on this information flow modelling approach in a

financial institution is shown by Stapel et al. in [46]. In [2], Allmann et al. and in [45], Stapel et al. further used it in the automotive industry to describe and improve the relationship between a car company (OEM) and its subcontractors.

Often, specific support tools can be built once an information flow problem has been identified, as discussed in [41]. The information flow and its presentation across solid and fluid allow to stimulate heuristic approaches that can be applied even before any given solid document exists.

7.3 Natural Language Processing in Requirements

Natural language is often used to support the specification of requirements, if only as an intermediate solution before formal modelling. As natural language is inherently ambiguous [5], several approaches have been proposed to automatically analyse natural language requirements to support requirements engineers for quality requirements specification documents [28,29,7,20]. Kof, Lee et al. work on extracting semantics from natural language texts by focusing on the semi automatic extraction of an ontology from a requirements document [28,29]. Their focus is on identifying ambiguities in requirements specifications. This ontology replaces a glossary and allows all stakeholders to communicate in a consistent way. This work may applicable for our context but not straight, This work may applicable to our approach, although not directly because i) Ontology Extraction remains a work-intensive task which needs to be integrated into the requirements engineering process, and ii) it does not support analysis per requirement and iii) it does not support the identification and refinement of security-relevant requirements.

Kiyavitskaya et al. describe ambiguity identification in natural language requirements specifications using tool support [21]. Their results partly apply to our approach, as both undetected ambiguous and security relevant requirements could cause severe problems during a project. However the ambiguity metrics presented in this work cannot easily be adopted to detect security requirements, but we agree that such tools should ideally have 100 % recall, not too much imprecision, and a high summarisation. This would allow the user to work on a set of potential ambiguous or security relevant requirements that is considerably smaller portion of the requirements specification. However if the recall is smaller, the user has to scan the whole specification for undetected requirements. As opposed to disambiguation, every requirement is to some degree security relevant. Therefore, in our context, the selection of some requirements is mainly a question of costs associate with refining it to security requirements.

Chantree et al. describe how to detect nocuous ambiguities in natural language requirements (i.e. how to interpret the conjunctions *and/or* in natural language) by using word distribution in requirements to train heuristic classifiers [7].

The process of creating the dataset is very similar to our work: collection and classification of realistic samples based on the judge of multiple experts to enhance the quality of the dataset. However, the heuristics are partly based on statistics from the British National Corpus (BNC). We did not find an obvious way to use such statistics for detection of security-relevant requirements. The reported results (recall = 0.587, precision = 0.71) are useful in the described context, but are too low for the SecReq approach.

8 Conclusion

In this paper, we addressed the problem that security becomes increasingly important in environments where there may not be any security experts available to assist in requirements activities. This situation leads to the risk that requirements engineers may fail to identify, or otherwise neglect, early indicators for security problems. We presented a tool-supported method that provides assistance for the labour-intensive and error-prone first round of identifying and analysing security requirements. The approach supports organisational learning on security requirements by establishing company-wide experience resources, and a socio-technical network to benefit from them. It is based on modelling the flow of requirements and related experiences. With help of these models, we enable people to exchange experiences about security-requirements while they write and discuss project requirements. At the same time, the approach enables participating stakeholders to learn while they write requirements. This can increase security awareness and facilitate learning on both individual and organisational levels. As a basis for our approach, we introduce heuristic assistant tools which support reuse of existing security-related experiences. The tool support makes use of a trained Bayesian classifier in order to heuristically categorise requirements statements as *security-relevant* resp. *less security-relevant*. We also showed how HeRA, our heuristic requirements assistant tool can be used to integrate that filter mechanism into a secure software development process. Note, that the approach is not restricted to HeRA, and can be used in other elicitation tools.

We evaluated this approach using several industrial requirements documents; ePurse, CPN, and GP. Our experiences with this "real-life" validation was overall positive: According to the numerical results, the approach succeeds in assisting requirements engineers in their task of identifying security-relevant requirements, in that it reliably identifies the majority of the security-relevant requirements (recall > 0.9) with only few false positives (precision > 0.8) in software evolution scenarios. Our evaluation of different training strategies shows that the classifier can quickly be adopted to a new domain when no previous versions of

requirements specifications are available for training. This could be done by a security expert during a first interview.

As another benefit of applying the FLOW approach in this context, it also the user to engage in "meta-learning", meaning that the approach supports the creation of experience regarding the learning process itself, which in turn can be used to improve the effectiveness of the learning process. As an outcome, this improvement can lead to better levels of security in practice through the intertwining learning of individuals and the organisation.

Our approach does not aim at completeness in a strict logical sense. There is no 100% guarantee that all security-relevant requirements are found, nor that no non-security-relevant requirements are falsely reported. This is, however, a limitation that is directly imposed by the current limitations from computational linguistics (essentially, the fact that a true automated text understanding is currently not available). In general, security experts cannot give such a guarantee, either. Therefore, we believe that the approach provides useful assistance in that it supports requirements engineers to identify security-relevant requirements, when no security expert is present. Even if security experts are present, our approach helps them to focus on already identified requirements and thereby efficiently use their limited time. Moreover, since this selection process is supported by automated tools, its execution is easy to document and it is repeatable and thus well auditable. This adds another level of trustworthiness to the process, compared to an entirely manual assessment.

Based on our work we see two main topics for future research. On the one hand, it would be interesting, if the approach could be applied to other types of cross-cutting quality requirements (e.g. safety or usability). On the other hand, the application in industrial practice will show the efficiency of our approach.

References

1. Christopher Alberts and Audrey Dorofee. *Managing Information Security Risks: The OCTAVE (TM) Approach*. Addison-Wesley, New York, USA, 2002.
2. C. Allmann, L. Winkler, and T. Kölzow. The Requirements Engineering Gap in the OEM-Supplier Relationship. *Journal of Universal Knowledge Management*, 1(2):103–111, 2006.
3. R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press, Addison Wesley, 1999.
4. B. Barber and J. Davey. The use of the CCTA risk-analysis and management methodology [CRAMM] in health information systems. In P. Degoulet, K.C. Lun, T.E. Piemme, and O. Rienhoff, editors, *MEDINFO '92*, page 1589–1593, North-Holland, 1992. Elsevier.
5. D.M. Berry and E. Kamsties. *Perspectives on Requirements Engineering*, chapter 2. Ambiguity in Requirements Specification, pages 7–44. Kluwer, 2004.
6. CEPSCO. Common Electronic Purse Specification (ePurse). http://web.archive.org/web/*/http://www.cepsco.com, accessed Apr 2007.

7. Francis Chantree, Bashar Nuseibeh, Anne de Roeck, and Alistair Willis. Identifying Nocuous Ambiguities in Natural Language Requirements. In *Proceedings of the 14th IEEE International Requirements Engineering Conference*, pages 56–65, Minneapolis, USA, 2006. IEEE Computer Society.
8. Lawrence Chung. Dealing with Security Requirements During the Development of Information Systems. In Colette Rolland, François Bodart, and Corine Cauvet, editors, *CAiSE*, volume 685 of *Lecture Notes in Computer Science*, pages 234–251. Springer, 1993.
9. D. Damian, S. Marczak, and I. Kwan. Collaboration Patterns and the Impact of Distance on Awareness in Requirements-Centred Social Networks. In *Proceedings of 15th IEEE International Requirements Engineering Conference (RE 2007)*, New Delhi, India, 2007.
10. Tom DeMarco. *Structured Analysis and System Specification*. Prentice-Hall, 1979.
11. F. den Braber, I. Hogganvik, M.S. Lund, K. Stølen, and F. Vraalsen. Model-based security analysis in seven steps - a guided tour to the CORAS method. *BT Technology Journal*, 25(1):101–117, 2007.
12. Paolo Giorgini, Fabio Massacci, and John Mylopoulos. Requirement Engineering Meets Security: A Case Study on Modelling Secure Electronic Transactions by VISA and Mastercard. In Il-Yeol Song, Stephen W. Liddle, Tok Wang Ling, and Peter Scheuermann, editors, *ER*, volume 2813 of *Lecture Notes in Computer Science*, pages 263–276. Springer, 2003.
13. GlobalPlatform. Global Platform Specification (GPS). <http://www.globalplatform.org>, accessed Aug 2010.
14. Siv Hilde Houmb, Shareeful Islam, Eric Knauss, Jan Jürjens, and Kurt Schneider. Eliciting Security Requirements and Tracing them to Design: An Integration of Common Criteria, Heuristics, and UMLsec. *Requirements Engineering Journal*, 15(1):63–93, March 2010.
15. International Standardization Organization. ISO 15408:2007 Common Criteria for Information Technology Security Evaluation, Version 3.1, Revision 2, CCMB-2007-09-001, CCMB-2007-09-002 and CCMB-2007-09-003, September 2007.
16. Neil Ireson, Fabio Ciravegna, Mary Elaine Califf, Dayne Freitag, Nicholas Kushmerick, and Alberto Lavelli. Evaluating machine learning for information extraction. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 345–352, Bonn, Germany, 2005. ACM.
17. Jan Jürjens and Pasha Shabalin. Tools for secure systems development with UML. *Int. J. Softw. Tools Technol. Transf.*, 9(5):527–544, 2007.
18. J. Jürjens. *Secure Systems Development with UML*. 2005.
19. E. Kelvin Kelloway and Julian Barling. Knowledge work as organizational behavior. *International Journal of Management Reviews*, 2:287–304, 2000.
20. Nadzeya Kiyavitskaya, Nicola Zeni, Travis D. Breaux, Annie I. Antón, James R. Cordy, Luisa Mich, and John Mylopoulos. Automating the Extraction of Rights and Obligations for Regulatory Compliance. In Qing Li, Stefano Spaccapietra, Eric Yu, and Antoni Olivé, editors, *Proceedings of 27th International Conference on Conceptual Modeling*, Lecture Notes in Computer Science, pages 154–168, Barcelona, Spain, 2008. Springer.
21. Nadzeya Kiyavitskaya, Nicola Zeni, Luisa Mich, and Daniel M. Berry. Requirements for tools for ambiguity identification and measurement in natural language requirements specifications. *Requirements Engineering Journal*, 13(3):207–239, September 2008.
22. Eric Werner Knauss. *Verbesserung der Dokumentation von Anforderungen auf Basis von Erfahrungen und Heuristiken*. Cuvillier Verlag, Göttingen, Germany, 2010. Phd Thesis.
23. Eric Knauss, Siv Houmb, Kurt Schneider, Shareeful Islam, and Jan Jürjens. Supporting Requirements Engineers in Recognising Security Issues. In Daniel Berry and Xavier Franch, editors, *Proceedings of the 17th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ '11)*, LNCS, Essen, Germany, 2011. Springer.
24. Eric Knauss and Daniel Lübke. Using the Friction between Business Processes and Use Cases in SOA Requirements. In *Proceedings of the 32nd Annual IEEE International Computer Software and Applications Conference (COMPSAC), Workshop on Requirements Engineering For Services*, pages 601–606, Turku, Finland, 2008.
25. E. Knauss and T. Flohr. Managing Requirement Engineering Processes by Adapted Quality Gateways and critique-based RE-Tools. In *Proceedings of Workshop on Measuring Requirements for Project and Product Success*, Palma de Mallorca, Spain, November 2007. in conjunction with the IWSM-Mensura Conference.
26. E. Knauss, D. Lübke, and S. Meyer. Feedback-Driven Requirements Engineering: The Heuristic Requirements Assistant. In *International Conference on Software Engineering (ICSE'09), Formal Research Demonstrations Track*, pages 587 – 590, Vancouver, Canada, 2009.
27. E. Knauss, K. Schneider, and K. Stapel. Learning to Write Better Requirements through Heuristic Critiques. In *Proceedings of 17th IEEE Requirements Engineering Conference (RE 2009)*, Atlanta, USA, 2009.
28. Leonid Kof. *Text Analysis for Requirements Engineering*. PhD thesis, Technische Universität München, München, 2005.
29. Seok Won Lee, Divya Muthurajan, Robin A. Gandhi, Deepak S. Yavagal, and Gail-Joon Ahn. Building Decision Support Problem Domain Ontology from Natural Language Requirements for Software Assurance. *International Journal of Software Engineering and Knowledge Engineering*, 16(6):851–884, 2006.
30. Raimundas Matulevicius, Nicolas Mayer, Haralambos Mouratidis, Eric Dubois, Patrick Heymans, and Nicolas Genon. Adapting secure tropes for security risk management in the early phases of information systems development. In Zohra Bellahsene and Michel Léonard, editors, *CAiSE*, volume 5074 of *Lecture Notes in Computer Science*, pages 541–555. Springer, 2008.
31. Daniel Mellado, Jesus Rodríguez, Eduardo Fernández-Medina, and Mario Piattini. Automated Support for Security Requirements Engineering in Software Product Line Domain Engineering. *Availability, Reliability and Security, International Conference on*, 0:224–231, 2009.
32. Daniel L. Moody. The “Physics” of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering. *IEEE Transactions on Software Engineering*, 35(6):756–779, Nov-Dec 2009.
33. H. Mouratidis, P. Giorgini, and G. A. Manson. Integrating Security and Systems Engineering: Towards the Modelling of Secure Information Systems. In Johann Eder and Michele Missikoff, editors, *CAiSE*, volume 2681 of *Lecture Notes in Computer Science*, pages 63–78. Springer, 2003.
34. Moussa Ouedraogo, Haralambos Mouratidis, Djamel Khadraoui, and Eric Dubois. An agent-based system to support assurance of security requirements. In *SSIRI*, pages 78–87. IEEE Computer Society, 2010.
35. M. Polanyi. *The Tacit Dimension*. Doubleday, Garden City, NY, 1966.
36. N. Russell, A. H. M. t. Hofstede, and W. M. P. v. d. Aalst. newYAWL: Specifying a Workflow Reference Language using Coloured Petri Nets. In *Eighth Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools.*, 2007.
37. Kurt Schneider and Daniel Lübke. Systematic Tailoring of Quality Techniques. In *World Congress of Software Quality 2005*, volume 3, 2005.

38. Kurt Schneider, Kai Stapel, and Eric Knauss. Beyond Documents: Visualizing Informal Communication. In *Proceedings of Third International Workshop on Requirements Engineering Visualization (REV 08)*, Barcelona, Spain, 2008.
39. Kurt Schneider. Software Process Improvement from a FLOW Perspective. In *Learning Software Organizations Workshop*, 2005.
40. Kurt Schneider. *Experience and Knowledge Management in Software Engineering*. Springer-Verlag, 2009.
41. K. Schneider. Generating Fast Feedback in Requirements Elicitation. In *Requirements Engineering: Foundation for Software Quality (REFSQ 2007)*, 2007.
42. D.A. Schön. *The Reflective Practitioner: How Professionals Think in Action*. Basic Books, New York, 1983.
43. G. Sindre and A. L. Opdahl. Eliciting security requirements with misuse cases. *Requirements Engineering Journal*, 10(1):34–44, 2005.
44. Kai Stapel, Eric Knauss, and Kurt Schneider. Using FLOW to Improve Communication of Requirements in Globally Distributed Software Projects. In *Workshop on Collaboration and Intercultural Issues on Requirements: Communication, Understanding and Softskills (CIRCUS '09)*, Atlanta, USA, November 2009.
45. K. Stapel, E. Knauss, and C. Allmann. Lightweight Process Documentation: Just Enough Structure in Automotive Pre-Development. In Rory V. O'Connor, Nathan Baddoo, Kari Smolander, and Richard Messnarz, editors, *Proceedings of the 15th European Conference, EuroSPI*, Communications in Computer and Information Science, pages 142–151, Dublin, Ireland, 9 2008. Springer.
46. K. Stapel, K. Schneider, D. Lübke, and T. Flohr. Improving an Industrial Reference Process by Information Flow Analysis: A Case Study. In *Proceedings of PROFES 2007*, volume 4589 of *LNCS*, pages 147–159, Riga, Latvia, 2007. Springer-Verlag Berlin Heidelberg.
47. TISPAN, ETSI. Telecommunications and Internet converged Services and Protocols for Advanced Networking (TISPAN); Services requirements and capabilities for customer networks connected to TISPAN NGN. Technical report, European Telecommunications Standards Institute.
48. Sholom M. Weiss and Casimir A. Kulikowski. *Computer systems that learn : classification and prediction methods from statistics, neural nets, machine learning, and expert systems*. M. Kaufmann Publishers, San Mateo, Calif., 1991.
49. S. Winkler. Information Flow Between Requirement Artifacts. In *Proceedings of REFSQ 2007 International Working Conference on Requirements Engineering: Foundation for Software Quality*, volume 4542 of *Lecture Notes in Computer Science*, pages 232–246, Trondheim, Norway, 2007. Springer Berlin / Heidelberg.
50. Alexander Wise. Little-JIL 1.5 Language Report. Technical report, Department of Computer Science, University of Massachusetts, 2006.
51. Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. *Experimentation In Software Engineering: An Introduction*. Kluwer Academic Publishers, Boston / Dordrecht / London, 2000.
52. *ST-Tool: A CASE Tool for Security Requirements Engineering*, Washington, DC, USA, 2005. IEEE Computer Society.
53. *SecTro: A CASE Tool for Modelling Security in Requirements Engineering using Secure Tropos*, London, 2011. CEUR-WS,vol - 734.