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 without expertise in security are at risk of overlooking security requirements, which often leads to security vulnerabilities that can later be exploited 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-relevant 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. They support reuse of existing experiences that are relevant for security. 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

Acknowledgements: This work was partially funded by the German National Science Foundation (DFG InfoFLOW 2008-2011) and the EU project Secure Change (ICT-FET-231101).

Noname manuscript No.
(will be inserted by the editor)

Enhancing Security Requirements Engineering by Organisational Learning

the date of receipt and acceptance should be inserted later

Abstract More and more software projects today are security-related in one way or the other. Requirements engineers without expertise in security are at risk of overlooking security requirements, which often leads to security vulnerabilities that can later be exploited 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-relevant 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. They support reuse of existing experiences that are relevant for security. 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.

Address(es) of author(s) should be given

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

1 Introduction

The growing complexity and interoperability of today's software systems creates new security challenges. Even components 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 partners. In order to close security loopholes, potential problems should be detected as early as possible during the development process. Requirements that may eventually affect system security need to be checked carefully before being implemented.

However, identifying those requirements is difficult: Complex business processes, organisational needs, and critical assets are handled by software systems. Thus, specifications from different project partners are voluminous and contain many requirements. Security requirements may be implicit, hidden, and spread out over different documents. For example, one requirement may call for easy web access; a different statement may require online shopping features. Taken together, both requirements are highly security-relevant. Any bug or unforeseen feature interaction in the systems can increase its vulnerability and diminish system security. It is tedious and error-prone to search a document manually or evaluate requirements during elicitation. Resources are usually limited for security analysis. Stakeholders often miss security-related requirements because of their limited security expertise and experience in assessing security implications. Thus, security issues are neglected and can cause substantial security problems later.

Unfortunately, potential threats cannot be identified once and for all, since threats to security are moving targets: 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. There are standards and best practices available aimed at guiding developers in building secure systems. Nevertheless, identifying requirements with security implications requires security expertise and experience. Unfortunately, security experts are not always available. It is, therefore, an organisational concern to encourage and support learning. From an organisational perspective, *one of the biggest problems in security engineering is the lack of security experts*.

We address this problem by reducing the dependency on experts by applying experience-based tools (e.g. the HeRA Heuristic Requirements Assistant [28], see Section 4). These tools set free expert resources. They can devote their attention to those tasks that still 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. [45]):

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 tools are enhanced in an integrated way.

In previous papers [17,30], intermediate *results* of this improvement effort were reported. The dedicated requirements engineering tools we developed include the HeRA heuristic requirements assistant [28] and its extension by Bayesian classifiers [17].

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: individual learning, infrastructure for exchange, and collection of expertise. We describe how our approach was applied to the requirements elicitation process at 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 telecommunications domain. It 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. However, our 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.

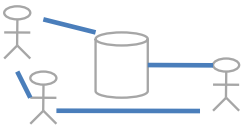
We use the security extension UMLsec [20] of the Unified Modeling Language (UML) for demonstrating how security requirements can be integrated in the system design phase. We show how experience from designing previous secure systems can be fed back into requirements elicitation activities. The UMLsec extension allows the system designer to include security requirements and other security-relevant information with models created using the UML notation. There are also tools for verifying the UMLsec models against the security requirements to ensure that the design supports the requirements [22,23,16]. UMLsec has already been validated in a number of industrial application projects, such as the one discussed in [21].

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. Section8 concludes the paper.

2 Organisational Learning on Security Requirements

Security experts are in high demand in many organisations 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.

Table 1 An 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 considers a concrete example used in this paper to illustrate the respective aspects of our approach.

Aspect	**Individual learning**	**Infrastructure for exchange**	**Collection of expertise**
Visualisation			
Challenges 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 documenting of requirements.
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 uses HeRA and Bayesian classifier like a 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.

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 get much more costly to handle if they are detected only later during implementation. We propose supporting 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 severe time pressure of a software project. How can individuals be encouraged and enabled to invest in learning while they work?
2. *Collection of expertise* is a key challenge: Security experts are already the bottlenecks in an 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 workflows, tools, and networks of people. 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? Most experience is tacit [41]. 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-relevant requirements which they might either have overlooked – or not considered a security issue.

Kelloway and Barling [24] 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 ex-

perts in some cases and mitigate their absence to some extent in other cases. At the same time, stakeholders can develop their skills in security requirements engineering.

Establishing a *collection of expertise* includes collecting experiences and requirements documents. It 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) [28]. HeRA allows to encode experiences as heuristic critiques based on a script language [29]. In addition, it is possible to capture knowledge about identifying security-relevant requirements using Bayesian classifiers [30]. 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. We discuss HeRA and tool-support for managing security requirements in more detail in Section 4.

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 [20] of the Unified Modeling Language (UML), which allows the designer to document security-relevant information as part of UML design models. It is, therefore, suited to document and transport security design experience. The UMLsec notation presented in [20] is only a core of a notation that is supposed to cover additional concepts as needed. It is supposed to be extended, and has been extended in the past.

Our elicitation tool HeRA assists stakeholders in specifying and analysing requirements with security implications. Above-mentioned experiences are reused and maybe evaluated for that purpose. 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 carried 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 [48].

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 considered incorrect or inappropriate, individuals may choose to change the respective heuristic rule. Heuristic rules automatically check whether critiques are applicable. We call the formalization of heuristics in those rules *encoding* of those heuristics, as seen in Figure 1). By encoding heuristics, the experience is added to the experience base.

In our case, it is important to include 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 [51]. We used the FLOW notation [44] for modeling the sequence of situations and improvements.

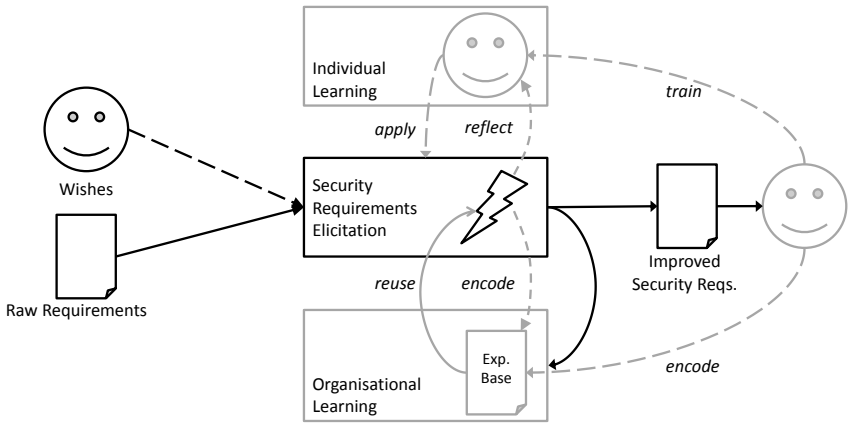


Fig. 1 Learning model, integrating individual and organisational learning.

FLOW was created as an information flow modeling language. It covers experience and both documented and undocumented (fluid) information, e.g. requirements [46,52]. FLOW syntax contains only a few simple symbols and arrows, as it is supposed to be used for discussions [38]. All elements will be introduced below as we need them to design improved flows. Notational syntax and semantics are explained in the legends of the FLOW models in this section. In [47], we compared the FLOW notation to a number of related modeling notations, such as data flow [10], process modelling [56], workflow [42], or UML. Those and other notations may be used instead of FLOW within our approach. The steps presented below illustrate that it was feasible and useful to use FLOW for that purpose.

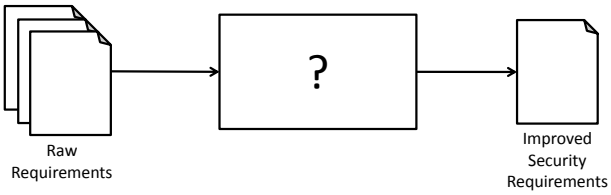


Fig. 2 Initial Situation: Raw requirements from several documents are transformed into an improved security requirements specification. The *document symbols* represent documented requirements or specifications. Three overlapping document symbols represent several documents of the same type. Raw requirements are transformed into improved security requirements during an *activity*. The activity is modeled by a rectangle. Since we know nothing about that activity yet, it is labeled with a question mark instead of an activity name. Arrows denote the *flow* of requirements.

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.

Analysis: The unstructured activity is treated like a black box. It 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, experts are often unavailable. 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* [18]. Common Criteria is an international standard (ISO/IEC 15408) for computer security certification. It consists of a collection of publicly available security domain experiences and guidelines. However, security expertise is required to understand the language used in the standard. Therefore, ETSI has derived understandable guidelines from Common Criteria. We consider them reusable 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 decomposed 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 measurable and testable security requirements through a number of steps, as shown in Figure 3. The refinement can be looked upon as a number of questions derived from the structure of ISO 15408. It helps stakeholder in refining and specifying security requirements.

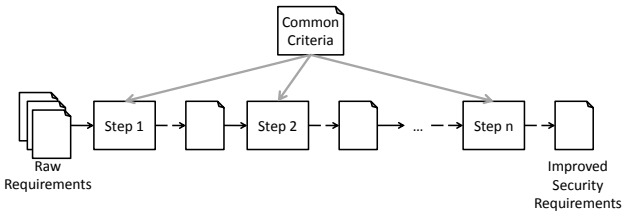


Fig. 3 Guidance by documented experience: Common Criteria guides the refinement of raw requirements into improved security requirements. A sequence of refinement steps provides experience from the Common Criteria for each step. In addition to the flow of requirements (black arrows), the availability 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 activities in steps 1 to n are being carried out. For example, the exact way of refining a requirement is being controlled by experience. The requirement itself is considered an input to this refinement activity. Input flow (e.g., requirements) reaches the activity rectangle from the side. Experience is attached to the top of the rectangle, which indicates that it is considered 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 requirements 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

Security experts are the main bearers of experience. This fact should be modelled 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.

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 or change. Resulting specifications still need to be integrated by an expert. The situation modelled 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.

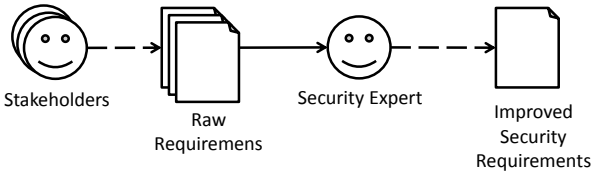


Fig. 4 Stakeholders write initial raw requirements. Requirements documents are written independently, resulting in several requirements documents. There are several stakeholders (e.g. representatives of customers or partners) and their documents, depicted by three overlayed symbols. All stakeholders are on their own. They create their respective documents independently of each other. The flow of requirements from those stakeholders to their respective documents has characteristic 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. We call information with these characteristics *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.

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.

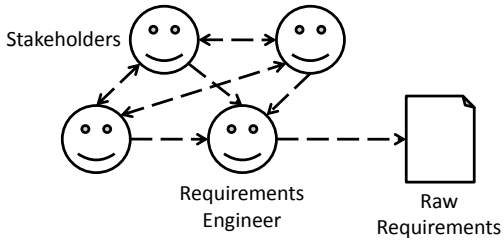


Fig. 5 Requirements are specified in a workshop. There are no new FLOW symbols in this model. The extensive use of dashed lines indicates the fluid nature of direct communication in a workshop. Depicting stakeholders separately allowed us to highlight the communication between them – which did not occur with isolated stakeholders as shown in earlier models.

Analysis: Talking is often considered faster and more convenient than writing a document and reading it. 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

spend a lot of his precious time with elicitation and capturing of ordinary requirements. Only a small subset is security-related.

3.5 Improvement: Elicitation Pattern with Tool

An important improvement was the introduction of tool assistance. HeRA is the Heuristic Requirements Assistant tool (see Section 4.1). It uses heuristic rules for scanning requirements and issues warnings when it detects potential problems. 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 requirements into HeRA. HeRA then analyses the requirement and gives feedback. Initially, this feedback is only based on generic knowledge, e.g. from textbooks. Experience feedback from the Security Expert can be added to HeRA during the project. Figure 6 depicts this scenario. In this figure the generic pattern has been applied to this HeRA situation.

The generic pattern consists of *someone experienced* (here: the security engineer) who invests experience, depicted as a gray dashed line to the top of the elicitation activity. Several *domain experts* (here: stakeholders) bring their domain knowledge to the meetings in an informal or fluid way. The experienced person gains *domain information* (here: requirements) and helps to document it. Elicitation in different contexts can use and instantiate that pattern. Fig. 6 shows its concrete application to eliciting security-relevant requirements.

Analysis: HeRA checks stakeholder requirements by means of 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 considerations 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 experience. They have been encoded in rules that can be automatically applied to requirements as they are written down. The focused discussions following a warning meet 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.

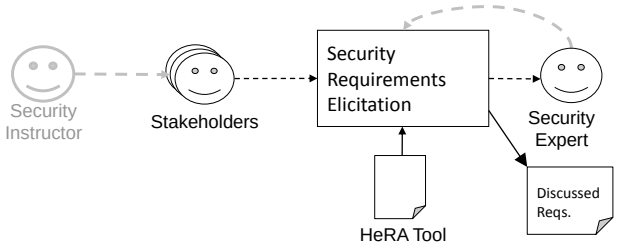


Fig. 6 Requirements Elicitation supported by a tool, with typical flows. The distribution of dashed vs. solid arrows characterizes the type of communication that determines this workflow. In this case, it is a careful mix of documented (solid) and fluid flows. Security instructors provide basic security 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 or specific requirements (black). HeRA is depicted as a solid part. It is connected to the activity rectangle from below. This indicates that HeRA supports the activity.

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. HeRA is solid and stimulates all those flows.

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. We called this approach of identifying security requirements *SecReq*.

Analysis: There are three parts of this model: The elicitation pattern on the left side feeds into the activity in the center. That activity refers to the above-mentioned five-step refinement strategy at ETSI (see [30]). 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. During construction, design decisions must be made. This is the point at which designers must consider security. They may gain new insights in the

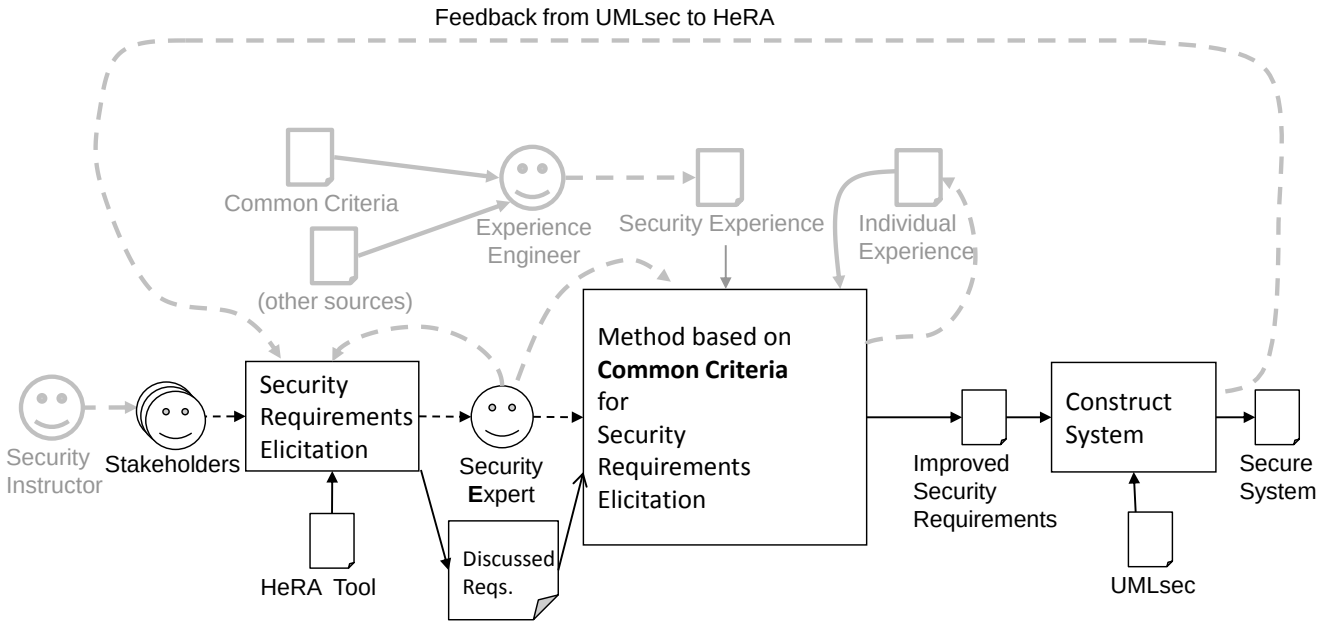


Fig. 7 The overarching flow infrastructure of our so-called SecReq model: HeRA supports the security requirements elicitation by offering rule-based critique. Downstream activities from construction feed back to inform security requirements elicitation. This figure describes the complete process and flow of security requirements. Although this figure is rather complex, it contains no new FLOW elements.

requirements that caused security considerations in design. This is the time to capture and map 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 *Construct System* to *Security Requirements Elicitation* in Figure 7 is fluid: Some expert or participant of system construction needs to take time formalizing the insight into a heuristic HeRA rule. Organisational learning challenges remind us that this altruism might not always occur under time pressure. Therefore, we envisioned a solid feedback flow that would cause no or very limited overhead.

We wanted to reuse documented experience. 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 were found. The key is to make those insights available to future projects – and to do that much earlier: 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 tried 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

Analysis: Bayesian classifiers are explained in Section 4. At this point, flow modelling is useless without implementing the tool features to support it. This model highlights that Security Requirements Elicitation will now be guided 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.

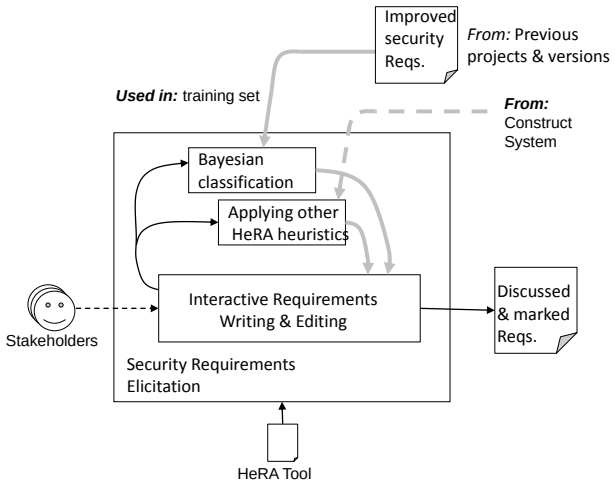


Fig. 8 Details of the activity *Security Requirements Elicitation* in Figure 7. Bayesian classifiers are envisioned to reuse feedback from requirements of previous projects. Some elements of this model have labels attached. They represent comments and are supposed to clarify the interfaces between this refined figure and the overall model in Figure 9 in Section 4.

Both feedback loops enhance organisational learning:

1. Explicit knowledge from designing secure systems is captured and formalized in HeRAs heuristics.
2. Classification knowledge is 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. In turn, 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 documents 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

In previous work we had developed the SecReq approach for elicitating and analysing security requirements [17]. It provides mechanisms to trace security requirements from high-level security statements, security goals, and objectives to secure design. 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 [18], (2) the HeRA tool with its security-related heuristic rules [28], and (3) the UMLsec approach for security analysis and design [20].

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. Security best practices such as standards ISO 14508 (Common Criteria) or 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 contributes along these lines. In particular, the approach aims at providing security best practises to mitigate the lack of security experience among developers. It offers a step-by-step security requirements identification, elicitation, and tracing approach to secure design [17].

The main goal of SecReq is to extend security requirements engineering process by seamlessly integrating elicitation, traceability, and analysis activities. The approach makes systematic use of the security engineering knowledge contained in the Common Criteria and UMLsec, as well as security-related heuristics in the HeRA tool. Security-related issues are identified from functional requirements and system descriptions using HeRA with its security-relevant rule sets which include guidelines and security-relevant keywords from Common Criteria and UMLsec. Common Criteria is also used to support security requirements refinement to testable and measurable expressions that can be realized in secure design as UMLsec design models. SecReq is tailored for non-security experts and also offers advice to developers when a security expert should be consulted. This way, the need for security experts is reduced to a need that better matches the resource available and makes the security part of a development project manageable. SecReq also enables reusing knowledge from project to project and learning from its success and failures by an explicit knowledge and experience feedback loop. 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.

HeRA (see Figure 10) is based on Fischer's architecture for domain oriented design environments (DODE) [12]. The central part of this architecture is a construction component. In the case of HeRA, requirements are "constructed" using a general-purpose requirements editor, a use case editor, or a glossary editor. These editors allow constructing the domain specific artifacts (i.e. requirements, use cases, and a glossary). HeRA offers two other DODE components, namely the Argumentation Component and the Simulation Component. Fischer [12] emphasizes the importance of arguing about hints from a domain-oriented design environment. An argumentation component allows users to adhere to warnings (and their respective rules), or to argue against them. Both types of feedback may lead to improved heuristics in the long term.

In HeRA, the argumentation component is based on *Heuristic Critiques*. These critiques consist of a heuristic rule, a meaningful message, and a criticality value. 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. In HeRA, we have currently two argumentation components in use. HeRA.Glossary analyses requirements and use cases

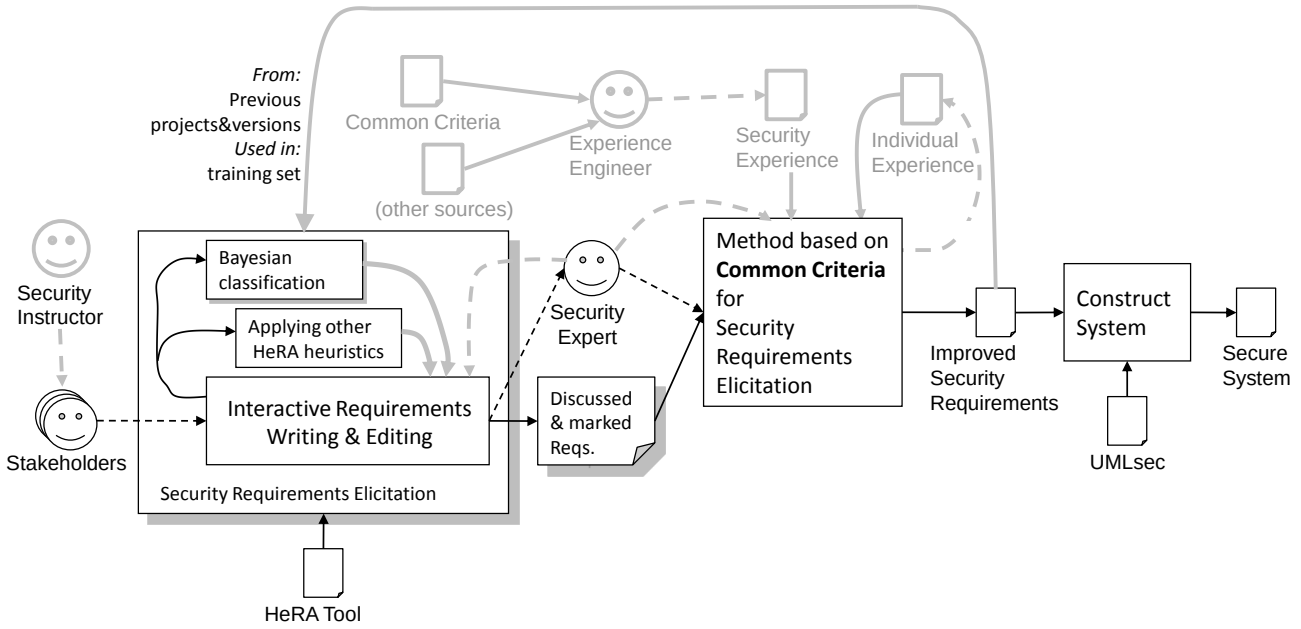


Fig. 9 Overview of SecReq approach with solid feedback from UMLsec to HeRA. In this model, Security Requirements Elicitation is supported by applying Bayesian classifiers. They are trained by reusing documents from earlier projects. Specifications from previous projects are used to train the classifiers.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

and recommends terms that should be defined in a glossary. Recommendations are based on the frequency of a given term. If a term is added to the glossary, HeRA.Glossary *learns* this term and recommends it in the next project with a higher priority. In HeRA.Glossary, the heuristic rules are hard-coded into the component. In contrast, HeRA.Critique has a library of heuristic critiques that can be adjusted by the user. The heuristic rules are encoded in Javascript. Any changes to a heuristic critique is directly applied. HeRA.Critique focuses on immediate, pro-active, and context-sensitive feedback. Even while the user edits requirements or use cases, the heuristic rules are evaluated in the background. Typical rules include checking for weakwords (e.g. "never", "someone"), consistency (e.g. each actor is listed as a stakeholder), and structure (e.g. all user-level use cases are referenced in a business-goal-level use case).

Heuristic critiques can be seen as an experience package with a strong focus on reuse [29]. Evaluation showed that *users write better requirements* documents with this kind of experience support [27]. Evaluation also showed that typical users are able to adjust existing rules or to create new heuristic critiques [32].

In contrast, the simulation component gives the requirements author feedback on the effects the current way of modeling use cases could have. As we want to use HeRA for the initial documentation of stakeholder wishes, we do not have a formal requirements model that could be simulated. However, we can derive certain models that give additional information about the requirements being documented.

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 [31]. Automatically derived use case point models help to *identify problems* like *requirements creep*, i.e. unperceived growth of demanded functionality over time [28].

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.

In the context of our SecReq approach, we use HeRA as it gives us the required infrastructure to analyse requirements very early and to visualise feedback to the author. Nevertheless, HeRA is a research demonstration tool. For practical use, central ideas of this research prototype should be transferred to professional requirements tools.

4.2 A Bayesian Classifier Extension for HeRA

In this paper, we describe an improved version of HeRA. Figure 10 shows a screenshot of editing security requirements in HeRA. On the left (1) the list of all requirements in the document is listed. Blue Icons highlight requirements that are classified as security relevant. In the center (2) re-

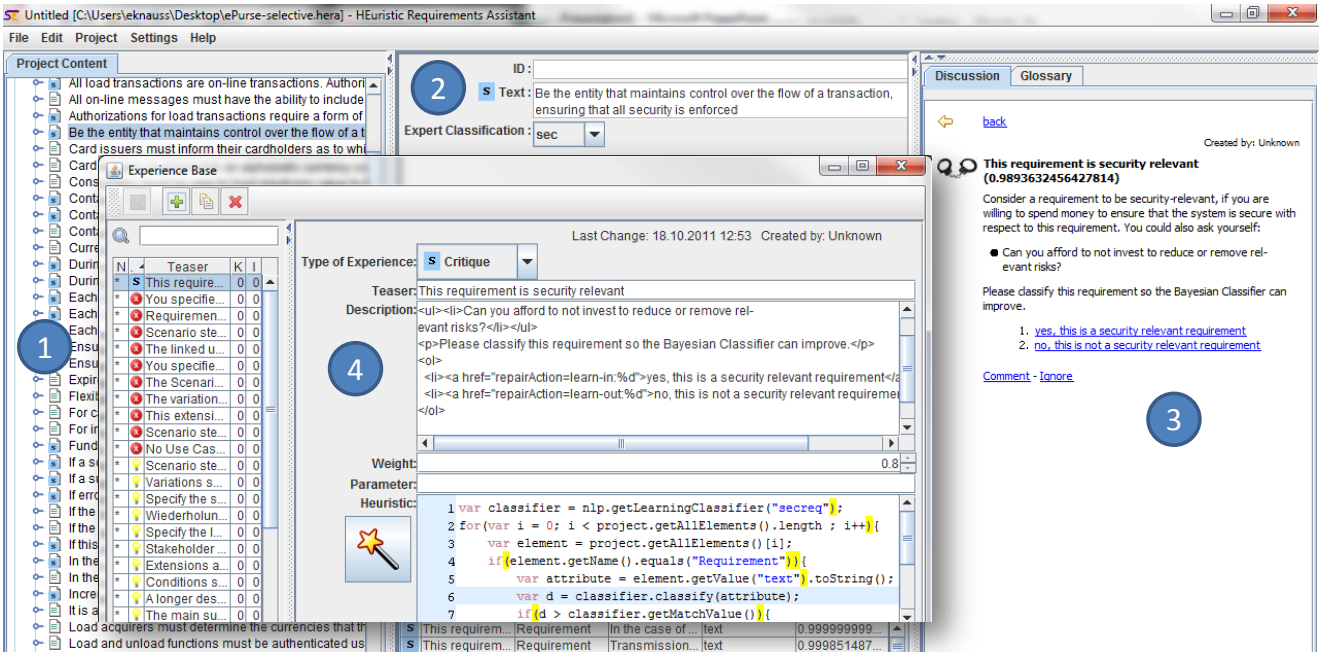


Fig. 10 A screenshot of editing security requirements in HeRA.

sides the construction component with an editor for requirements. *Text* is the currently edited requirement, expert classification is the classification of our experts. Again, an icon highlights the field our classifier found a security relevant requirement in. On the right site (3) is the argumentation component. It allows the user to argue about the feedback, by training the Bayesian classifier, commenting on the feedback, or to ignore the critique. It is also possible to adjust the heuristic critiques in HeRA's Experience Base (4). The user can easily change the type of the critique (i.e. error, warning, hint, information, or (new) security critique). HeRA will show warnings only if no errors exist. The teaser defines the heading displayed in the argumentation component (3). The description defines a longer explanation of the feedback and (new) the actions to train the classifier. The weight represents priority: HeRA will display up to three critiques of a type with the highest weight. The heuristic rule can have parameters (such as keyword lists). The rule itself is encoded as Javascript. This particular rule initialises a Classifier and than iterates over all requirements, classifying them. If a match is found, it is added to the list of context-sensitive feedback.

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 classifier. Growing numbers of pre-classified requirements from previous projects will increase the classification ability of that classifier. 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.

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:

> *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.
> *Example:* "The card account balance should not be modifiable by unauthorized parties." (the security requirement of *integrity*).

In our context, security-relevant aspects may not explicitly refer to security issues, but affect security.

> *Security-relevant requirement:*
> *(i)* A requirement that is a refinement of 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 and which is not classifed as a *security requirement* or *security-relevant requirement*.
Example: "The card and the PSAM must use a public key algorithm for mutual authentication and session key exchange [...]"

Security requirements, *security-relevant requirements*, and *security-related requirements* are closely related. In our example above, the relation would be:

- In order for the card account balance not to be modifiable by unauthorized parties, it must be ensured that a Cancel Purchase transaction is always performed by the same POS device as was used for the purchase being canceled. (refinement of *security requirement* to *security-relevant requirement*)
- To ensure that a Cancel Purchase transaction is always performed by the same POS device as was used for the purchase being canceled, one can use public key algorithms for mutual authentication and session key exchange (refinement of *security-relevant requirement* to *security-related requirement*).

Note that the concepts of *security requirements*, *security-relevant requirements*, and *security-related requirements* are categories in the realm of informal concepts. Thus, there cannot be a entirely formal distinction between them as if they were expressed in formal logic.

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: Confidentiality and Integrity 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 is:

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

For our purposes, we believe these two classification questions to be essentially equivalent. That is, we believe that the risks identified using the first question are the same as those identified by the second. However, it would be interesting future work to investigate whether this is indeed the case.

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 [47] which was designed for modelling flows of requirements and experiences. However, other notations might also be used for that purpose. The sequence of models [47], analyses [51], and conclusions [52] demonstrates the reflective process involved with improving infrastructures [50]. 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, Bayesian Classifiers will be evaluated in depth. They are the latest addition to the tool.

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 5.1.3. Finally, we show and discuss the results for each evaluation goal in Sections 5.1.4, 5.1.5, and 5.1.6.

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 classifiers can be transferred to other domains.
(G3) Evaluate how useful practitioners consider automatically identifying security requirements.

We will evaluate goals (G1) and (G2) in the next section in detail. 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 should be carried out in the future.

5.1.2 Obtaining Testdata

For our evaluation, we need several sets of already classified requirements. We use one part of these sets for training and the other part for assessing the performance of the classifier. For our purposes, there are two main sources for classified requirements available: (i) Expert classification and (ii) Requirements databases.

Expert classification: The most obvious approach is to let experts classify a set of publicly available requirements. We consider a requirement to be *security-relevant* if three out of four experts consider it to be security-relevant based on our classification question (see Sect. 4.2). A requirement is not *security-relevant* if three out of four experts say so. Other requirements are considered unclear and not suitable for classification and evaluation. The advantage of this approach is that we get a conceptually clean classification, with good repeatability. Disadvantages are the high effort associated with this approach, as well as the fact that our experts have not always worked in the projects we use the requirements of. This lack of domain knowledge makes it sometimes hard to answer the classification question. We used *expert classification* on the publicly available specifications of the Common Electronic Purse (ePurse) [6] and the Global Platform (GP) [15].

Requirements databases: The second approach is directly obtaining classified requirements by using databases from past projects. In this case, a requirement is *security-relevant*, if it is associated (e.g. via tracing link) to a *security requirement*. This corresponds to our classification question: Indeed, money was spent to refine this requirement in order to make the system secure.

The advantage of this approach is that if a suitable requirements database exists, this could produce large training and evaluation sets with low effort. The disadvantage is that it is hard to find a good and comprehensive requirements database with consistent tracing links that can be used to analyse the security requirements. Either the effort will be high to reproduce the tracing links, or the security issues will be sensitive. We were able to classify requirements from the Customer Premises Network specification (CPN) in this way [53].

For the goals (G1) and (G2) we used both *expert evaluation* and a *requirements database*. 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). The last column gives the source of the classification (last column).

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.

5.1.3 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, 19]. This validation method ensures that statistics are not biased for a small set of data [54]. The dataset is randomly split into k parts of equal size to achieve a uniform distribution of security relevant requirements among the k parts. $k-1$ 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].

Table 2 Industrial requirements specifications used for evaluation.

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

Based on the data reported in [19], we consider f-measures over 0.7 to be good. For being useful in our SecReq approach, recall should not be lower than 0.7. Given the problem of missing security expertise, our approach could still be useful when missing 30% of security relevant requirements. If it drops even lower, automatic identification would barely be beneficial. For our purpose, high recall is considered more important than high precision. Therefore, we would allow the precision to drop as low as 0.6, if the recall can be improved accordingly: Missing a security relevant requirement is worse than scanning a view irrelevant.

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**

5.1.4 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 11, 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 11 helps to evaluate whether the training set is sufficient. The most typical training curve can be observed with the GP data: Adding more training data leads to high improvement in the beginning, and only low improvements later. With the ePurse data, it is different: Even the baseline (i.e. selecting all requirements to be relevant) is almost good enough for our purposes, because of the high percentage of security relevant requirements in that dataset. CPN data leads also to an untypical linear training curve. As this data was obtained from a database, it is possible that the classification is not as strict as our expert classification. Thus, new training data adds always the same amount of new information, leading to an almost linear course. Apart from the training, results exceed the above-mentioned thresholds for recall and precision. Hence, we consider the classifier useful.

5.1.5 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 classifier cannot easily be used in a different context.

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

we get poor results if we apply a Bayesian classifier trained with a specification from one domain to a different domain (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 classifier

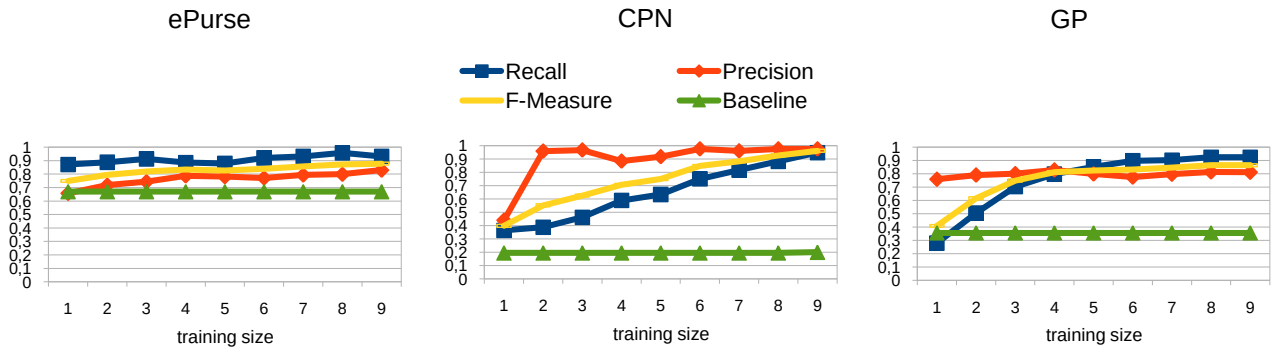


Fig. 11 Results of 10-fold cross validation using only one specification. Baseline is the precision we get when classifying all req. to be security-relevant.

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 obtained good results for all three specifications included in the evaluation. Figure 12 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 11, 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.

6 Discussion and Implications on Industrial Practice

In Section 5, we evaluate the possibility of identifying security-relevant requirements by using a Bayesian classifier. It is important to note that we used the Bayesian classifier differently during evaluation and productive application (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 refinement wizard that allows laypersons to start with the refinement themselves.

In this section, we discuss whether the observed results are sufficient for employing the classifier in practice. Then, we take a look at the validity of our evaluation of the Bayesian classifier. Finally, we summarise the discussion with practitioners and describe how they perceive the implications of the classifier 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 obtained 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. Still, this classifier is not as good as the ones specific for a specification. This could imply that a good classifier should not be diluted with data from a different domain. However, the dilatation factor seems to be small according to our data. In addition, the training curve indicates potential, implying that a general classifier for security relevance can be created by applying considerably larger training sets and specifications from more domains.

To summarise, the classifier in its current status is indeed a very valuable addition for example in the context

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	recall	0.91	0.95	0.85	0.88
+ GP	precis	0.79	0.80	0.94	0.78
	f-m.	0.84	0.87	0.89	0.83

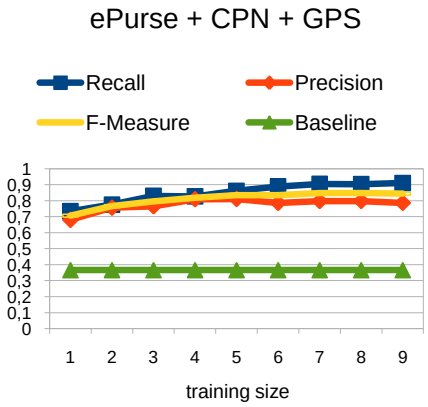


Fig. 12 10-fold cross validation, multiple training

of software evolution or product lines. Hence, the classifier could be trained using the last version of the requirements specification and than 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 11 apply to this situation. In other situations, the learning curve in Figure 12 and tests with systematic training with falsely classified requirements show that the classifier quickly adopts to new domains.

6.2 Discussion on Validity

Wohlin et al. define types of threats to validity for empirical studies [57]. 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 quality of the evaluation process, the evaluation goals and the distribution of evaluation variables (indirect and direct variables).

In our case, assumptions made on the classification question and our criteria for good results are critical for determining the quality 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. 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 classifier. Ideally, we would train the classifier 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 deal with 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) 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), also 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 separates important from less 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 13 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.

Advantages:

- No expert needed for training, no instruction needed
- Up-to date experience can be used – the same as for productive work

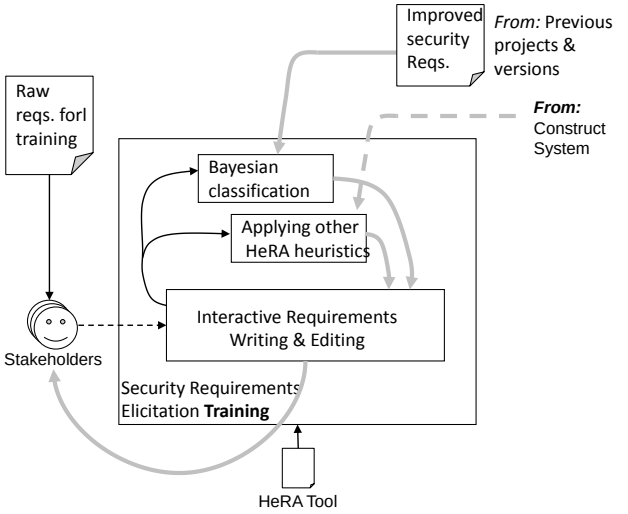


Fig. 13 A stakeholder trains security requirements elicitation.

- 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 [39, 13]. 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 [14]. 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 [35]. 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. present an agent-based system to support assurance of security requirements [40]. 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. present an approach which adapts Secure Tropos for security risk management in the early phases of information systems development [36]. 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 [49]. Mellado et al. present the *SREPPLine tool* which provides automated support for the security requirement engineering process for software product lines (SREPPLine) [37]. 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 [23] supports the analysis of the security aspects expressed in the security extension UMLsec [20] 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 [55]. 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 [47]. 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 [55], 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 [51]. In [2], Allmann et al. and in [50], 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 [43]. 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 [33, 34, 7, 25]. 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 [33, 34]. 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 [26]. 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, the selection of some requirements is mainly a question of costs associate with refining it to security requirements in our context.

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.

The approaches discussed above relating to security requirements engineering are important. But the works do not adequately focus on the issues which put real challenges during the elicitation of security requirements. In particular, from an organisational perspective one of the biggest problems in security engineering is the lack of security experts. Our research aims at addressing this problem by reducing the dependency on experts. The proposed approach facilitates the security requirements elicitation process, by incorporating organizational learning through modelling the flow of requirements and related experiences to the security requirements engineering. We believe this work on the one hand enables project practitioners to exchange their experiences about security-requirements while analysing project requirements and on the other hand increases security awareness and facilitates learning within both individual and organisational levels.

8 Conclusion

Security is of increasing importance in industry. Even in environments with few or no security experts, emerging products may turn out to be security-relevant. There is often a lack of security experts who can assist in requirements activities. Overlooked and neglected indicators for security issues in early requirements can cause severe problems later.

The approach suggested in this paper supports establishing company-wide experience resources, a socio-technical network, and an infrastructure that encourages individual learning. Thus, it creates benefits as an integral part of organizational learning.

The HeRA Heuristic Requirements Assistant supports reuse of security-related experiences. We demonstrated the use of trained Bayesian classifier for heuristically categorising requirements statements as *security-relevant* or *less security-relevant*, respectively. We described how HeRA and its classifying mechanism were integrated into a secure software development process. Other elicitation tools could replace HeRA in this context and flow of experience. By feeding improved and classified requirements into UMLsec, software construction benefits directly from the increased awareness and improved input.

We evaluated this approach using several industrial requirements documents. According to the above-mentioned numerical results, the approach succeeded in assisting requirements engineers: The majority of security-relevant requirements produced only a few false positives. It is a crucial task to identify security-relevant statements early for in-depth analysis and security considerations. The classifier could be adopted quickly 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.

Due to the heuristic nature of our elicitation approach, there is no 100% guarantee for finding all security-relevant requirements. There will always be false positives, too. On the one hand, this limitation stems from the limitied ability to understand natural language texts in computational linguistics. On the other hand, human experts are not able to identify security-relevant statements perfectly, either. We are confident our approach will provide useful assistance to requirements engineers who have no security experts at hand. Even those experts can themselves benefit from an automated pre-screening based on previous experience. Our approach with its defined flow of information is easy to document, repeatable, and, thus, well auditable.

We have modeled and designed the flow of requirements and experiences using FLOW. This notation and technique

for information flow modeling was originally developed to improve existing software processes. Documented and informal communication were visualized and discussed with process owners and participants in order to resolve bottlenecks and anomalies in communication.

In this paper, FLOW has been used pro-actively: We conceived the flow of solid and fluid experiences before the constituent parts (e.g., HeRA and UMLsec) had been combined before. This paper shows some of the evolutionary steps from an unknown black-box requirements activity to a fine-grained model of interrelated flows. FLOW has succeeded in stimulating research and integration of results in this case. We plan to apply this technique to other situations where top-down planning of communication and bottom-up development of tools are to be conceived.

Currently, our approach was optimized for ETSI as an organization with a constant throughput of security-relevant projects. The step-wise activity of requirement analysis in SecReq reflects this fact. In future work, additional sources of security experience and knowledge should be tapped. The static input from Common Criteria and a security expert could be opened up to include more dynamic sources of experience, e.g. from other business units, companies, or even universities. In a globalized world, the inclusion of external knowledge and experience can be a facilitator for applying this approach to different organizations and smaller companies, too.

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. Gerhard Fischer. Domain-Oriented Design Environments. *Automated Software Engineering*, 1:177–203, 1994.
13. 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.
14. Paolo Giorgini, Fabio Massacci, John Mylopoulos, and Nicola Zannone. ST-Tool: A CASE Tool for Security Requirements Engineering. In *RE '05: Proceedings of the 13th IEEE International Conference on Requirements Engineering*, pages 451–452, Washington, DC, USA, 2005. IEEE Computer Society.
15. GlobalPlatform. Global Platform Specification (GPS). http://www.globalplatform.org, accessed Aug 2010.
16. Sebastian Höhn and Jan Jürjens. Rubacon: automated support for model-based compliance engineering. In Robby, editor, *ICSE*, pages 875–878. ACM, 2008.
17. 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.
18. 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.
19. 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.
20. J. Jürjens. *Secure Systems Development with UML*. Springer-Verlag, 2005.
21. J. Jürjens, J. Schreck, and P. Bartmann. Model-based security analysis for mobile communications. In *30th Intern. Conference on Software Engineering (ICSE 2008)*. ACM, 2008.
22. J. Jürjens and G. Wimmel. Formally testing fail-safety of electronic purse protocols. In *16th International Conference on Automated Software Engineering (ASE 2001)*, pages 408–411. IEEE Computer Society, 2001.
23. Jan Jürjens and Pasha Shabalin. Tools for secure systems development with UML. *Int. J. Softw. Tools Technol. Transf.*, 9(5):527–544, 2007.
24. E. Kelvin Kelloway and Julian Barling. Knowledge work as organizational behavior. *International Journal of Management Reviews*, 2:287–304, 2000.
25. 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.
26. 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.

27. 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.
28. 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.
29. E. Knauss, K. Schneider, and K. Stapel. Learning to Write Better Requirements through Heuristic Critiques. In *Proceedings of 17th IEEE Requirementes Engineering Conference (RE 2009)*, Atlanta, USA, 2009.
30. 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.
31. 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.
32. Eric Werner Knauss. *Verbesserung der Dokumentation von Anforderungen auf Basis von Erfahrungen und Heuristiken*. Cuvillier Verlag, Göttingen, Germany, 2010. Phd Thesis.
33. Leonid Kof. *Text Analysis for Requirements Engineering*. PhD thesis, Technische Universität München, München, 2005.
34. 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.
35. S. Islam M. Pavlidis. SecTro: A CASE Tool for Modelling Security in Requirements Engineering using Secure Tropos. In *CAiSE '11: Proceedings of the CAiSE forum 2011*, pages 89–96, London, 2011. CEUR-WS,vol -734.
36. Raimundas Matulevicius, Nicolas Mayer, Haralambos Mouratidis, Eric Dubois, Patrick Heymans, and Nicolas Genon. Adapting secure tropos 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.
37. 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.
38. 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.
39. 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.
40. 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.
41. M. Polanyi. *The Tacit Dimension*. Doubleday, Garden City, NY, 1966.
42. 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.
43. K. Schneider. Generating Fast Feedback in Requirements Elicitation. In *Requirements Engineering: Foundation for Software Quality (REFSQ 2007)*, 2007.
44. Kurt Schneider. Software Process Improvement from a FLOW Perspective. In *Learning Software Organizations Workshop*, 2005.
45. Kurt Schneider. *Experience and Knowledge Management in Software Engineering*. Springer-Verlag, 2009.
46. Kurt Schneider and Daniel Lübke. Systematic Tailoring of Quality Techniques. In *World Congress of Software Quality 2005*, volume 3/3, 2005.
47. 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.
48. D.A. Schön. *The Reflective Practitioner: How Professionals Think in Action*. Basic Books, New York, 1983.
49. G. Sindre and A. L. Opdahl. Eliciting security requirements with misuse cases. *Requirements Engineering Journal*, 10(1):34–44, 2005.
50. 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.
51. 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.
52. 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.
53. 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, 2010.
54. 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.
55. 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.
56. Alexander Wise. Little-JIL 1.5 Language Report. Technical report, Department of Computer Science, University of Massachusetts, 2006.
57. 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.