Formative user-centered evaluation of security modeling: Results from a case study

Sandra Trösterer
University of Salzburg, Austria
Elke Beck
University of Salzburg, Austria
Fabiano Dalpiaz
University of Trento, Italy
Elda Paja
University of Trento, Italy
Paolo Giorgini
University of Trento, Italy
Manfred Tscheligi
University of Salzburg, Austria

ABSTRACT
Developing a security modeling language is a complex activity. Particularly, it becomes very challenging for Security Requirements Engineering (SRE) languages where social/organizational concepts are used to represent high-level business aspects, while security aspects are typically expressed in a technical jargon at a lower level of abstraction. In order to reduce this socio-technical mismatch and reach a high quality outcome, appropriate evaluation techniques need to be chosen and carried out throughout the development process of the modeling language. In this article, we present and discuss the formative user-centered evaluation approach, namely an evaluation technique that starts since the early design stages and actively involves end-users. We demonstrate the approach in a real case study presenting the results of the evaluation. From the gained empirical evidence, we may conclude that formative user-centered evaluation is highly recommended to investigate any security modeling language.

Keywords: Security modeling, usability, formative evaluation, user-centered development, security requirements engineering

INTRODUCTION
Modeling languages are fundamental elements for any software engineering methodology. They are used to capture requirements, design systems, and specify desired properties. However, the role of security modeling is not yet clearly defined in mainstream software engineering. As pointed out by Devanbu and Stubblebine (2000), a major challenge in security software engineering is the unification of conceptual abstractions, methodologies and tools differently adopted in security and system engineering.

In SRE (McDermott & Fox, 1999; Sindre & Opdahl, 2005; Giorgini, Massacci & Mylopoulos, 2003), security concerns are considered since the beginning of the engineering process along stakeholders' needs and constraints. Here, unifying security with system aspects is inherently more complex than in later phases of the software development process. System aspects are expressed and modeled at a *social/business* level, so to allow mutual understanding between requirements engineers and stakeholders. On the other hand, security aspects are expressed by security experts at a *technical* level. SRE modeling languages are,

thus, very likely to suffer from a *socio-technical mismatch* that prevents an effective conduction of such development phase.

Even for expert designers, accommodating orthogonal perspectives and minimizing the socio-technical mismatch can be a very complex task. This calls for specific evaluation techniques involving end-users throughout the development of a security modeling language. Thus, the choice of adequate evaluation techniques, a sound set-up of the evaluation study, and an accurate analysis of the results are essential to assess a security modeling language.

Many evaluations of (security) modeling languages exist in literature, e.g. Opdahl and Sindre (2009), Recker et al. (2009), and Kärnä, Tolvanen, and Kelly (2009). However, such studies were conducted after the language was adopted (summative evaluation). To the best of our knowledge, there are no publicly available evaluations of security modeling languages prior to the language release. In this paper, we report about our experience with formative evaluation of a security modeling language, called STS-ml (Socio-Technical Security modeling language), which is currently under development in the context of the EU-sponsored project Aniketos[i]. STS-ml is a security requirements engineering modeling language expressly thought to model and analyze security concerns for composite services (Casati, Sayal & Shan, 2001). STS-ml is being developed according to an iterative development paradigm where internal releases are followed by evaluation studies.

The objective of conducting a formative evaluation before the language is released, is to gather feedback from a variety of end-users (requirements engineers, security experts, and domain experts), so to refine the language in the subsequent development iterations. This way of considering the needs of end-users throughout the development process is known as user-centered design approach. Our evaluation focuses on the adequacy of the modeling primitives for expressing security concerns as well as an initial assessment of the usability of the language and its support tool. In order to identify weaknesses of both the language and its support tool, we focused the evaluation around the following research questions:

- How usable are the modeling language and its support tool?
- Are there missing concepts that would be essential to model security aspects?
- Is the graphical representation adequate / easy to understand?
- Are there concepts whose semantics is unclear / underspecified?
- Are there technical issues that limit the usability of the tool?

These questions can be reformulated in terms of the three main usability criteria to be described in the baseline section. Questions regarding the adequacy of the representations or missing concepts primarily focus on the *effectiveness* of the security modeling language, whereas the understandability of the representation, the easiness of fulfilling modeling tasks, and the helpfulness of the language and its support tool are aspects to be considered when it comes to *efficiency*. Users' *satisfaction* can be seen generally in terms of positive and negative comments by the users, their like/dislike of the language and its support tool, but also annoying technical problems.

While the specific evaluation results are language-dependent, the lessons learned are valid for security modeling languages for SRE. As we will show in the following sections, formative evaluation sessions are beneficial to the design of security modeling languages, as they contribute both to mitigate the socio-technical mismatch and to early identify usability issues that affect the user acceptance of the language and its support tool.

The rest of the paper is structured as follows. First we present our baseline: the formative user-centered evaluation approach. Then STS-ml, the language that we use as a case study, will be introduced. We will then describe the method followed during the evaluation workshop. In the further sections we present the obtained results, analyze and discuss them, and finally we present our conclusions, ongoing work, and future research directions.

BASELINE: FORMATIVE USER-CENTERED EVALUATION APPROACH

In Human-Computer Interaction (HCI), the purpose of evaluation is to assess and assure the quality of a designed artifact for its users. In this paper, such artifact is a tool-supported security modeling language for composite services (Dalpiaz, Paja & Giorgini, 2011). Our evaluation focuses on the usage of the language and its support tool, instead of assessing the quality and correctness of the created models. According to Dix et al. (2003), evaluation has three main goals: "to assess the extent and accessibility of the system's functionality, to assess the users' experience of the interaction, and to identify any specific problems with the system" (p. 319). To achieve these main goals, various evaluation approaches can be applied. In this section, we discuss the characteristics of the formative user-centered usability evaluation approach, which we applied in the case study, and thus we are able to position our approach within the variety of methods to evaluate modeling languages and tools. In the following, we will detail each characteristic of the evaluation approach and explain why we selected this specific approach.

Evaluation criteria: Usability as a general quality criterion

The evaluation process is typically guided by a pre-defined collection of quality criteria or user requirements. Our applied evaluation approach takes usability as a general quality criterion for evaluation. Usability is defined as the extent to which users achieve their goals with effectiveness, efficiency, and satisfaction (ISO 9241-11:1998). To guide the evaluation of the tool-supported modeling language, we need to further detail the notions of effectiveness, efficiency and satisfaction.

Effectiveness is about how accurately and thoroughly users are able to achieve their specified goals with a designed artifact. With regard to the effectiveness of modeling languages and tools, effectiveness means that users are able to achieve all their modeling goals. To effectively create models, users need an adequate, i.e. complete and clear, set of language components (notation semantics), which comprises all concepts users need during their modeling activity. Moreover, the graphical notation (graphical symbols) for the notation semantics needs to be unambiguous and complete. According to Moody and van Hillegersberg (2009), this principle of semiotic clarity and completeness contributes to cognitive effectiveness of modeling languages.

The second usability measure – *efficiency* – is related to the resources expended for effectively achieving a goal. Modeling languages and tools shouldn't require the users to spend more resources (e.g. mental effort, time) than justifiable to achieve their modeling goals. For instance, how quickly a user is able to learn the language and create a model, depends on the cognitive load the modeling language notation poses on the user (Moody & van Hillegersberg, 2009). Furthermore, the efficiency of a modeling activity is also related to the degree to which the modeling language is properly supported by a tool (Hommes & van Reijswoud, 2000).

Finally, *satisfaction* of users is an important aspect as well, and it is defined as the extent to which users are free from discomfort, and which attitudes they have towards an artifact. It is the subjective impression of the users about the modeling language and tool. Such impression can, for instance, be assessed by checking positive or negative comments expressed by the users.

Many other quality characteristics of modeling languages and tools could be considered as evaluation criteria (e.g. Paige, Ostroff & Brooke, 2000; Nysetvold & Krogstie, 2005; Kahlaoui, Abran & Levebvre, 2008). For instance, one may consider the consistency of all parts of the modeling language with its intended purpose, or the scalability of the modeling language (its capacity to model both small and large systems). We chose to focus on usability in our evaluation approach because, since the STS-ml and its tool are in an early development

stage, such criterion is the one we can evaluate with more accuracy. Other criteria, such as scalability and consistency, would require a more mature version of the language/tool.

Evaluation goal: Formative evaluation for iterative refinement

Evaluation approaches can be classified into formative and summative evaluation (Rosson & Carroll, 2001; Hix & Hartson, 1992; Scriven, 1967). The main difference between formative and summative evaluation is the goal the evaluation pursues.

Summative evaluation is concerned with the measurement of the overall quality of a final design artifact. It (usually) takes place at the end of the development process to assess how well the requirements are met.

Formative evaluation aims to identify problematic aspects in the design artifact when the design/development process is not finished yet and no final design artifact is available. The insights gained from the formative evaluation process serve as feedback for the designers of the artifact.

A range of evaluation studies has applied a summative evaluation approach for assessing the quality of (security) modeling languages and tools. For instance, several experimental or metrics-based comparisons of different modeling languages and methods (e.g. Opdahl & Sindre, 2009; Recker et al., 2009; Kärnä, Tolvanen & Kelly, 2009) are conducted in the tradition of summative evaluation. By contrast, the formative evaluation approach has received less attention when it comes to evaluating modeling languages and tools. To the best of our knowledge, moreover, there are no publicly available studies that analyze formative evaluation with respect to security modeling languages.

We chose to apply formative evaluation for investigating the usability of the STS modeling language and tool because this approach enables us to explore usability problems in a very early phase in the design process of the language and tool. It has the advantage to indicate problematic design decisions while the design process is still ongoing and thus to correct the trajectory of design activities. Moreover, this approach also helps to identify possible ways to address the identified usability problems.

Role of users within evaluation: Participation in evaluation activities

Besides their contribution to defining the evaluation goal, several perspectives on the role of users within evaluation are available when following a User-Centered Design (UCD) approach (see ISO 9241-210:2010). In UCD, human-centered activities are included throughout the development life cycle by focusing on users during the planning, design and evaluation of an artifact. This focus on users in evaluation may be understood in different ways, leading to different roles of the user in the evaluation process:

- The user can be considered in evaluation studies by defining behavioral models of (categories of) users. Such models are then used to assess usability;
- The user can be a principal (a stakeholder), whose interests are known by an expert. The expert then acts as an advocate of the user within evaluation studies;
- The user may be a participant and, thus, plays an active role in the evaluation studies.

These three roles of users within evaluation are visible in model-based, expert-based, and user-based usability evaluation methods (Dillon, 2001), respectively. Each method requires different techniques for studying and evaluating cognitive complexity (comprehensibility) of software modeling languages (Kamandi & Habibi, 2008). Each of the user roles in evaluation methods and techniques comes along with specific benefits and drawbacks for evaluation.

In our case study, we invited actual (or future) end-users of the STS modeling language and tool to actively participate in our evaluation activities. Users are all stakeholders who are, or are going to interact with the modeling language and tool, and they are seen as domain experts with rich knowledge and valuable experience. Their experience is gathered by asking them to use the STS-ml support tool and to communicate their feedback to us. Taking user

interests and needs into account since the early stages contributes to improving the acceptance and perceived usefulness of the language and tool.

Evaluation methods variety: Method triangulation

When applying formative evaluation, assessing the reliability and validity of the applied methods in the notion of quantitative research is not possible. An alternative approach is triangulation (e.g. Golafshani, 2003). The validity of the evaluation results is more grounded when data is triangulated, i.e. collected via different methods, measures, approaches, and when the analysis and interpretation of these data lead to similar conclusions (Wilson, 2006). Thus, for the evaluation of STS-ml and its tool, different methods are applied to investigate one issue (i.e. the usability of the modeling language and tool), and the findings from the differently collected data are compared to identify similarities and differences. In contrast to applying a single method for answering research questions, having more data sources at hand is helpful to come up with a deeper understanding of the collected data.

CASE STUDY: DEVELOPMENT OF A TOOL-SUPPORTED SECURITY MODELING LANGUAGE

The main purpose of the EU-sponsored Aniketos project is that of creating and dynamically maintaining trustworthy and secure composite services. A composite service (Casati, Sayal & Shan, 2001) consists of a bundle of services that, together, provide service consumers with a composite functionality. There are standard languages to define composite services (e.g. BPEL4WS[ii]); however, security aspects are not supported in a standard manner. The main purpose of STS-ml is to enable the specification of security and trustworthiness requirements for composite services. The current version of STS-ml focuses mostly on security modeling, with a very limited support for trustworthiness requirements.

STS-ml (Dalpiaz, Paja & Giorgini, 2011) builds on top of Tropos (Bresciani et al., 2004) and its security-oriented extension (Giorgini et al., 2005). It is founded upon the notion of actors, service providers and consumers, which interact by offering and consuming services, respectively. These actors are generally mutually independent and, therefore, their behavior is unknown and non-controllable. Thus, the best a composite service designer can do is specifying constraints on their interactions. STS-ml reflects such intuition and relates security requirements to interaction.

We exploit the notion of *social commitment* to express security requirements – in a domain-independent way – so to regulate the interaction among actors. Commitments are promises with contractual validity that actors make and get from one another, to achieve their own objectives. Formally, commitments are a quaternary relation C(debtor, creditor, antecedent, consequent) between a debtor and a creditor (both being actors), in which the debtor commits to the creditor that, if the antecedent is brought about, the consequent will be brought about. A service interface is a set of commitments the provider makes to the prospective consumers that the service will be delivered as described by the interface.

Figure 1 outlines STS-ml: the specifications of security requirements for the composite service are derived once the modeling is done, and the constraints (security needs) imposed by the actors are expressed by the modelers.

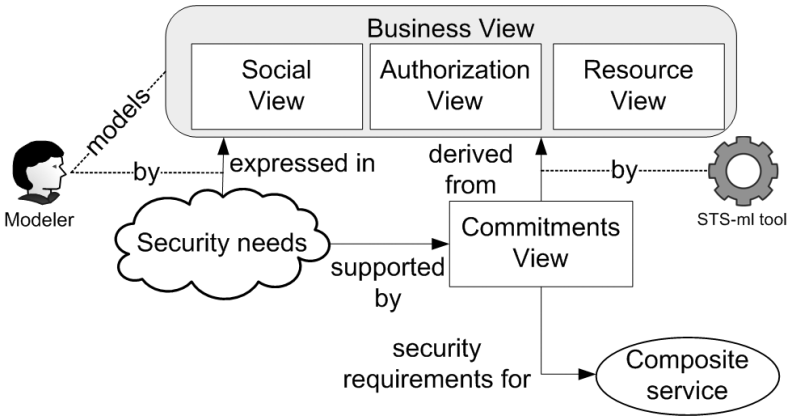

Figure 1. Outline of the STS-ml; from security needs to security requirements

STS-ml supports multi-view modeling: interactions among actors can be represented by focusing on different perspectives (views) at a time. As shown in Figure 1, STS-ml consists of three different views: social, authorization, and resource. The security needs are expressed in the business view, which consists of the three aforementioned views. The business view is automatically mapped by the STS-ml tool to the commitments view, which supports the security needs expressed in the business view, and is the specification of security requirements for the composite service.

We summarize the three views of STS-ml: the social view, resource view, and authorization view; more details can be found in (Dalpiaz, Paja & Giorgini, 2011):

- The *social view* represents actors as *intentional* and *social* entities. Actors are intentional as they have goals they want to achieve, and they are social because they interact with others to get things done, mainly by delegating goals or exchanging information;
- The *resource view* gives a structured representation of the resources in the given setting. We consider resources to be intangible, referring to information irrespective of its representation, or tangible, referring to the actual representation of the information by some support means;
- The *authorization view* shows the permission flow from actor to actor, that is the authorizations actors grant to others about information (intangible resources).

Currently STS-ml supports these security needs: (i) *non-repudiation* of a delegated goal; (ii) *redundancy* requires a service provider to adopt redundant strategies to deliver a service; (iii) *no-delegation* requires a service provider not to delegate to others the delivery of a service; (iv) *non-disclosure* requires that some information is not transferred to other actors; (v) *need-to-know* requires the usage of resources only for a certain purpose; and (vi) *integrity* requires a resource is not modified in an unauthorized way. The first three security needs are constraints over actors' social interactions (in the social view) whereas the other three relate to the exchange of information (in the authorization view).

Computer-aided support: the STS tool

The STS tool serves as computer-aided support for the STS-ml; modelers can use the STS tool to create STS-ml diagrams. The tool allows for the creation of diagrammatic models; for each view presented in the description of the STS-ml, a separate diagram is created using the tool, allowing the modeler to focus on one view at a time. Security needs are specified over the models. Future versions of the tool will provide automated reasoning support to identify consistency issues as well as security and trust concerns. Technically, the STS tool is an Eclipse RCP application[iii] built upon the GMF framework[iv] to create graphical meta-model based editors. Figure 2 shows how the tool looks like in its improved version when modeling

the social view of a scenario, in which a tourist wants to organize a trip using a Travel Planner service.

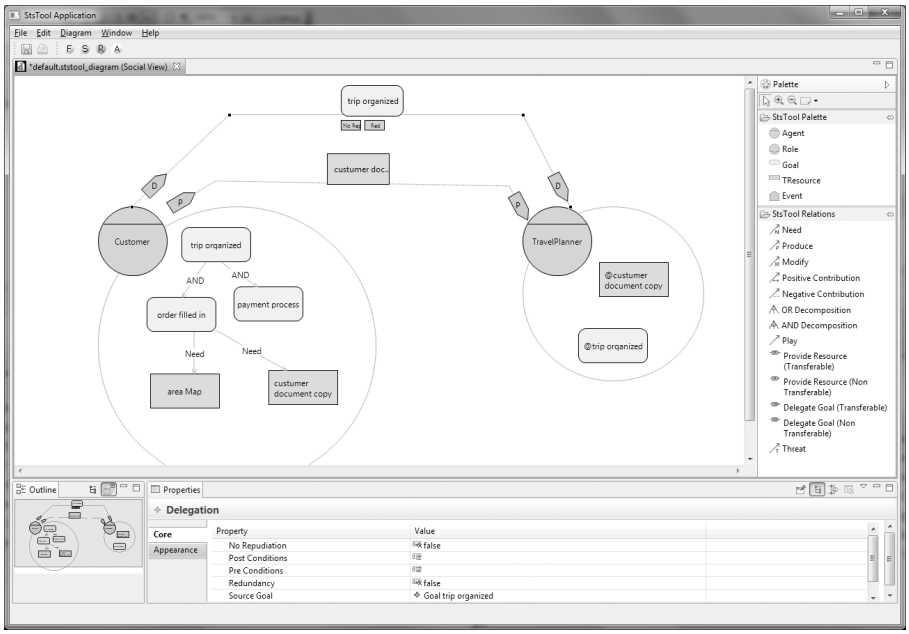

Figure 2. STS tool, showing the social view of an STS-ml diagram (improved version)

EVALUATION WORKSHOP

In order to investigate the research questions mentioned in the introduction, an evaluation workshop was scheduled for one and a half days, beginning with a presentation of the modeling language and tool on the first day, and practical work with the language and tool on the second day. Different evaluation methods (questionnaires, observation, interviews, group discussion) were adapted to the setup of the workshop. As three participants could not participate in the workshop physically, the setup was also especially adapted to involve them remotely.

Two people were present to supervise the evaluation. Three members involved in the design of the STS-ml and tool supported the modeling exercise on the second day. Two of them gave presentations about the modeling language and tool on the first day of the workshop.

Participants

In total, seven application domain experts from three different companies participated in the workshop and worked with the modeling language and tool. Demographic data is available from 6 participants (3 male, 3 female) as one person missed to fill out the corresponding online questionnaire. The participants had a mean age of 32 years. The youngest person was 29, the oldest 37. Four participants had experience in system modeling with UML, and one of them was experienced in security modeling with Si* (Massacci, Mylopoulos & Zannone, 2010) and UMLSec (Jürjens, 2002) as well. All others, though accustomed to reading and analyzing models, did not have system or security modeling experience so far.

Procedure

An outline of how our evaluation study is structured is depicted in Figure 3. On the first day of the workshop, all participants were welcomed and were given a presentation about the modeling language and the modeling tool, each lasting about 90 minutes. The remote participants could follow the presentations via web-based conferencing services. During the

presentations, participants were allowed to ask questions at any time. Whereas the part on the modeling language was a presentation, participants had the opportunity to do some exercise with the modeling tool on a predefined scenario during the presentation of the modeling tool. After each session, participants were asked to fill out an online questionnaire about the quality of the training, and at the end, after both sessions, they had to fill out another online questionnaire about their gained knowledge of the modeling language and tool at that point.

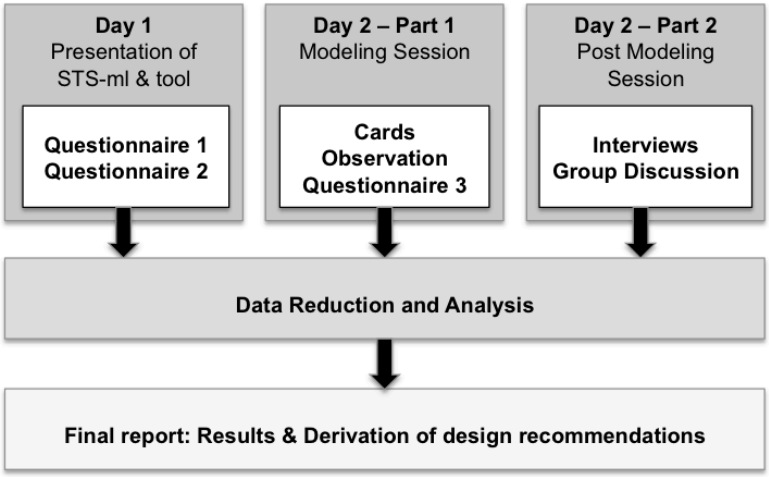

Figure 3. Outline of our formative evaluation study

For the modeling session, which took place in the morning of the next day and lasted approximately 3 hours, participants were first instructed that their task would be to model a predefined scenario (case study) from their application area. The three areas comprise (A) telecommunication, (B) air traffic management and (C) e-government. Two participants modeled two scenarios from case study A, two further participants a scenario from case study B, and 3 participants a case study C scenario. Each group included at least one expert of the modeled domain. Participants were informed that for the purpose of evaluation they will be provided with some materials to work with during the modeling session, and that the purpose for this will be to give them the opportunity to make notes, write down comments, and rate the concepts and graphical representations of the modeling language and tool at any time during the modeling process. Therefore they got a questionnaire concerning the adequacy of the concepts and graphical representations of the modeling language and tool to fill out during the modeling session, and a set of cards, with the instruction to fill out a card whenever they have the impression that a concept is missing, or they experience a serious problem during the modeling process. This instruction was given verbally and in written form.

Each modeling group had a dedicated room, where a laptop with an attached 22-inch monitor (without "touch" functionalities) and a mouse were provided. Additionally a camera was positioned in an angle right behind the participants to record audio and video of the screen and possible gestures of the modelers (see Figure 4). The three remote participants were informed in advance about the technical requirements and settings for the online conferencing services that were used to communicate in the session.

Additionally, all participants had already been informed that they would be video- and audio-recorded during the modeling session, and were first asked to sign a consent form. Then the video recording was started and the modeling group was observed from a distant position by an evaluator, except the remote participants, who were just video-recorded at this point. An STS-ml designer was present in case participants had severe problems and could not continue modeling without help. The evaluators had previously instructed the designers about the allowed interaction protocol with modelers. Then each modeling group started their task.

Figure 4. Camera position for the recording of the modeling session

After the modeling session was finished, each modeling group was interviewed for approximately 30 minutes. In the interview, participants were asked about their experience with the modeling language and tool. Finally a group discussion, lasting about 90 minutes, was conducted with all participants and five designers, moderated by an evaluator.

Evaluation methods: Data collection and analysis

In the workshop, we pursued the method triangulation approach in order to gain a high validity of the evaluation results. With regard to our research questions we chose different appropriate data collection methods for usability evaluation. The questionnaires given to the participants primarily addressed specific aspects like e.g. the adequacy of the graphical representations, whereas the methods interview, observation, and group discussion contribute to the answer of the research questions in multiple ways. In detail, the following methods were chosen:

Method 1: Questionnaires and cards. Participants were given three questionnaires in total, one that dealt with the quality of the training on the first day, one that focused on the knowledge of the modeling language and tool after the training and a third one, which could be filled out during the modeling session, was focusing on the concepts and graphical representations of the modeling language and tool. Additionally, to give the participants the opportunity to make notes about occurring problems or missing things about the modeling language and tool, they were given a set of prepared cards at the beginning of the modeling session, which they could fill out anytime during the modeling. Those cards were blank, expect for a short headline ("Problem / Missing Concept – Short description") and a slot to fill in the time when the problem occurred.

Questionnaire 1 (quality of training) was an available questionnaire that was adapted to an online version within the Aniketos project. It had to be filled out twice by the participants, once after the presentation of the modeling language, and once after the presentation of the modeling tool. Participants had to answer questions concerning the training materials, the trainer, training arrangement, overall satisfaction, and pacing on a 6-ary scale. The questionnaire was primarily given to the participants in order to assess whether the training was successful from their subjective perspective. Indeed, a training which is perceived as not useful could negatively affect the evaluation results.

Questionnaire 2 (knowledge) was an online questionnaire, which participants filled out after the training on the modeling language and tool. The questionnaire served the purpose to capture objectively the degree of knowledge participants had about the modeling language and tool before the modeling session began. They were asked about their modeling experience, age and gender, and were presented an image of the social, the resource, and the authorization view of the example model used during the training, respectively. In order to access the knowledge of the participants, sentences were presented for each view, which describe the model in natural language. Participants were asked to indicate for each sentence whether they think the sentence is true or false, and how confident they are about their decision. 10 sentences in total had to be judged by the participants. Additionally, if they discovered comprehension difficulties in the security modeling language, they were asked to freely describe these difficulties.

Questionnaire 3 (concepts and graphical representations) was given to the participants at the beginning of the modeling session and could be filled out any time during the session. They were thereby provided a list of all concepts of the modeling language with its corresponding graphical representation. For each concept participants should answer whether the concept is *well-defined*, *useful* and whether the graphical representation is *adequate*. If any criterion was not full-filled, participants should write down the problem in keywords.

Method 2: Observation. The method of observation was chosen to identify problems that participants might not be aware of during the modeling process, and to identify issues that occur on a more abstract, process level. The observer thereby had a prepared guideline with questions he/she should especially focus on during the observation. Apart from noting start, end, and break times, the observer had to keep track of the discussions between the modelers, questions directed to the developers in the room, or if any interventions from the developers happened. Additionally, the focus lay on the things that made the modelers "stop" their task. In order to allow cross validation (and as backup option) the sessions were also video-recorded from the observer's viewpoint.

Method 3: Structured interview. The main purpose of the interview was to gain a deeper understanding of the participants experience with the modeling language and tool. The interview questions were prepared, comprising questions e.g. regarding the overall experience, what participants liked or disliked, if there were any parts of their scenario that were difficult or impossible to model (due to limitations in the security modeling language), or if the tool provided all necessary functions and information.

Method 4: Group discussion. The purpose of the group discussion was to collect, categorize, discuss, and prioritize the problems that have been identified during the workshop by the participants. Therefore, the cards that had been filled out during the modeling session were collected and arranged in categories on a whiteboard, and a 3-dot query (a common moderation technique) was used to identify the most urgent problems, i.e. each participant had three dots that could be distributed on three problems he or she would find the most important ones.

Method 5: Data analysis methods. The application of a mix of different data collection methods during the workshop also required a special way of analysis. Quantitative data gained from the questionnaires was analyzed statistically. Qualitative data was analyzed as follows: comments and answers from the interviews were shortened, categorized, and complemented by the observations made during the modeling session and the notes participants made on the provided cards (including a categorization of this input as well). This

method was chosen in order to gain the best overall picture. In sum, approximately nine hours of video and audio data material were produced, which had to be transcribed and analyzed.

RESULTS

In this section, we present selected results of the evaluation workshop in a condensed form, thereby focusing on the most important aspects and implications for the STS-ml and tool. Generally speaking, problems concerning the underlying concept of the modeling language, functionalities of the language (analysis, views), usability of the tool and the modeling process itself (interpretation) were identified. In sum over 50 core problems and/or issues could be extracted.

Generally speaking, participants were satisfied with the quality of training. Regarding their knowledge about the STS-ml and support tool nobody of the participants did assess all the sentences presented in questionnaire 2 right. The highest number of correct judgments where the participant was definitely sure about his or her decision was 9 (out of 10 sentences to assess). The mean was 7. It shall be considered, however, that the training subject was complex, and that it was the participants' first training on the STS-ml and tool.

General aspects

These aspects might be regarded as relevant for any security modeling language, as they comprise a more general set of problems:

Training. Participants noted that the modeling language is quite complex, and the skills level and used terminology of the modelers might differ. This issue confirms our claim about the socio-technical mismatch in security modeling languages (for SRE): on the one side, modelers need to correctly depict the business/social aspects of the setting; on the other side, they need technical knowledge about security properties and methods. A good training is an essential prerequisite. The provision of adequate training materials (e.g. presentations, tutorials/help in the tool) should be therefore incorporated by all means.

Problems with the interpretation. It was unclear for the participants whether there is a correct way of modeling, and how they can be sure if what they are doing is right. Participants had problems to identify what should be part of the model, how the scenario description can be interpreted, or how to identify the right goals. It turned out that "there is a big interpretation gap" from the scenario to the model description. This is especially an important point when it comes to the applicability of the final model at a later stage. Aspects that were not considered or integrated during the modeling process will decrease the representative quality of the model and therefore its applicability. Moreover, since STS-ml is a security modeling language, aspects that are omitted result in a possible security breach, as security analysis has not been comprehensive.

Aspects specifically concerning the modeling language

Missing security needs / trust relations. Participants noted that, at the moment, several security needs (e.g. risk perception, threats/hazards, dependability) as well as trust relations are not considered in the language. As a consequence, the language does not allow for relating security and trust aspects, which are intertwined in many settings.

Missing concept of time / process aspects. So far, changes over time (before – after) or occurring processes (cause – effect) are not considered in STS-ml. It would be useful to represent that in a way that depending on the states of particular resources, the behavior of an actor changes. Another requested feature is to specify that an actor has to carry out tasks in a certain order.

Definition of concepts. Some concepts are not sufficiently defined at the moment (e.g. produce vs. modify resources, Authorization (Transferable) vs. Authorization (Non Transferable)), and a clearer separation and/or definition should be provided for those concepts.

Delegation prohibition. The possibility to explicitly prohibit the delegation from one actor to another is missing.

Resources with alternative contents. Resources can have alternative contents, and participants wondered whether it is possible to have an OR relationship for resources as well.

Shared resources. Participants wondered specifically how to model shared resources, i.e. resources having multiple owners (e.g. a bank account).

Analysis provided by the language and tool. It was not clear for the study participants what kind of analysis the tool can support in an automatic way, and whether it is meant to support risk analysis. It was also unclear whether currently unknown threats will be discovered using the tool, and whether it can be used in an exploratory way.

Views. Participants mentioned that the three views might not be sufficient for all application domains. Depending on the application domain, the organization, or the target user, an additional view or other views could be helpful. Giving modelers the possibility to create/build custom views could overcome this problem.

Adequacy of the graphical representations. Participants made some comments on the adequacy of the graphical representations. Generally, the size, color, and/or filling of certain representations needs to be adjusted. It was also noted that some graphical representations are too similar and should be differentiated more clearly.

Aspects specifically concerning the modeling tool

During the evaluation workshop, several usability issues concerning the modeling tool were identified. The participants suggested improvements regarding how to draw connection lines, changing the type of elements on-the-fly, and a list of all elements in a diagram. Apart from identifying usability issues to be fixed urgently in order to provide a proper functionality and usability of the tool, further aspects could be identified that might be especially helpful if models become larger and more complex:

- Auto-save of files
- Automatic layout functionalities to rearrange elements in a diagram
- Zoom in/out on the model
- Reduction of the Graphical User Interface (GUI) complexity depending on the current work (just show a palette of features that are relevant for the current task/view in the tool)
- History of changes, which can be commented. This would be helpful to understand certain modeling steps after a long time (and also for evaluation purposes)
- Library of patterns/possibility to save substructures of the model. It could be time-saving and helpful for the modeling process to model small scenarios and then build the entire picture
- Online help within the tool
- Simple views of the model that provide a high-level outline

DISCUSSION

The formative evaluation of the STS-ml and its support tool has successfully identified problems and suggestions to be considered in the further development of the language and tool. Not only could problems be identified, but also possible solutions were depicted and discussed by the participants.

As regards our research questions we could find that certain concepts are missing or are not sufficiently defined at the moment, that there are several usability issues concerning the tool, and that the graphical representations need some adjustments in terms of clearer differentiation and illustration. Some of the identified issues are very specific to the STS-ml and its support tool, while others can be seen as general problems to be considered in the design of any security modeling language (especially for SRE).

Socio-technical mismatch. The challenge that we outlined in the introduction was empirically evidenced. Many of the identified issues are, indeed, related to the difficulty in representing both, the social perspective (how actors operate and interact), and technical aspects that regulate their interaction from a security standpoint. The obtained feedback went well beyond the simple confirmation of such issue; participants did suggest possible ways to alleviate the mismatch, both in terms of modeling primitives and of methodological guidance.

Importance of training. Training emerged as a feature of utmost importance to enable effective usage of a security modeling language. Such training shall be designed so to support users with different backgrounds and skills. Indeed, a security modeling language can potentially be used by a variety of users, among those software engineers, requirements analysts, security experts, domain experts, risk analysts, and system architects.

Subjective interpretation/modeling. The problem that end-users may have a different interpretation of a model, and that they typically model the same scenario in different ways is a challenge to be considered in the development process. From a methodological viewpoint, the question is whether there are ways to provide guidance throughout the modeling process apart from former training, e.g. online help or hints provided by the modeling tool. In general, the results clearly evidence that there are important issues to be addressed in further iterations of the development process so to ensure good applicability.

Scalability. The scenarios used in the workshop were not extremely complex; we foresee that, as the size/complexity of the model increases, problems related to scalability (e.g. lack of space, automatic layout of elements) will become crucial, and new problems might arise. During the workshop, participants did proactively express wishes and suggested possible workarounds concerning scalability, e.g. the possibility to save certain structures of the model, to have a library of patterns, to easily zoom in/out of the model, to switch between overview and detailed views, to have a history of changes, and the possibility to add comments to such changes.

Triangulation pros and cons. Some usability issues/problems (7% of all issues/problems) were depicted with almost every applied evaluation method, whereas we could also see that more subtle aspects could be just brought up using a specific method. 44% of the usability issues and problems were solely identified via the interviews, 24% only with the use of cards, and observing the modeling activities discovered 15% of all usability issues/problems. From that point of view, we can certainly say that our method triangulation approach provided more all-embracing results than the application of just a single method. Moreover, having usability problems iteratively described and discussed – for instance, problems noted on the cards

during the modeling session were discussed in the group discussion later on – helped us in understanding the usability problems more thoroughly. However, regarding the efficiency of the method triangulation approach it has to be pointed out, that such an approach also leads to a high effort of time and skills when it comes to data analysis. Especially the structuring and condensing of the qualitative data is time-consuming and needs expertise in qualitative research processes. On the other hand, the improvements and adjustments of the language and its support tool that can be done at this early stage based on the evaluation results certainly legitimate the effort. One has to take into account at this point that with progressing development of a security modeling language it becomes increasingly difficult to make changes of the conceptual basis, i.e. it is easier to fix problems detected at an early stage compared to problems that are detected in the final development phase.

Prioritizing issues. Due to the large amount of issues that the evaluation allowed to gather, prioritization techniques become essential to determine which problems to solve first and which issues have minor relevance. The group discussion was especially useful to define a first-approximation priority for the challenges by outlining a set of issues to be solved urgently. We classified problems according to four types: major problems that require major effort to fix them, major problems that require minor effort to fix them, minor problems that require major effort to fix them and minor problems that require minor effort to fix them. Different problem categories shall be addressed in different ways: major problems shall be fixed to make the language applicable; minor problems let language/tool designers estimate the required effort and decide on how to proceed. Ideally, though, all problems should be addressed.

CONCLUSIONS & OUTLOOK

In this article, we reported about a successful application of a formative user-centered evaluation approach to identify problems and weaknesses of a security modeling language and its support tool at an early design stage. The contribution of the study is twofold. First, we empirically confirmed our intuitions about the socio-technical mismatch that affects security modeling, one of the reasons why security modeling is an extremely complex task. Second, we could identify and analyze a large amount of usability problems related to the support tool and the adequacy of the graphical representations.

End-users involved in the study have actively contributed not only by reporting issues, but also by suggesting workarounds and possible solutions. The results of the evaluation – including both problems and suggestions – were given back to the designers of the STS-ml as a final evaluation report. In the report the results of the evaluation were presented at a fine-grained level and conclusions and recommendations for improvements were made.

Based on the evaluation results, several issues have been and are being solved. STS-ml and its tool support new concepts: an event element and a “threat” relation starting from events to represent hazards, pre- and post-conditions assigned to various elements to express temporal relations. The naming and semantics of some concepts is under (re-)definition, e.g., intangible resources and transferrable/non-transferrable authorizations. To improve scalability, all goals related to an actor can be hidden, and zoom-in/out functions have been added. A number of bugs were fixed in the tool, and a bug-tracking repository was set up to classify and systematically address reported bugs.

Our empirical results clearly suggest that formative user-centered evaluation is highly recommendable when it comes to the development of security modeling languages. It contributes to ensuring that the requirements of the language are met, concepts are adequate, and important aspects are not missing. The effort for such an evaluation is considerable; however, the degree to which language and tool can be improved makes it worth the effort.

Formative evaluation is an approach that is meant to accompany the entire development process. Therefore, further work will focus on the evaluation of improved versions of the STS-ml and its support tool to make sure that the made improvements are adequate and the general requirements are still met. This will also include adjustments of the evaluation methods; for instance, the method will be tuned to assess whether modeling larger scenarios yields further issues to be analyzed. In our evaluation the modelers were application domain experts. In further evaluations, other end-users like requirements engineers or security experts shall be involved in order to gain further feedback from different perspectives.

So far we focused on usability as core evaluation criterion. In further investigations of more advanced versions of the STS-ml and its support tool we also plan to focus on modeling patterns to enhance *scalability,* or techniques to ensure *consistency* (e.g. among views).

REFERENCES

Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., & Mylopoulos, J. (2004). Tropos: An Agent-Oriented Software Development Methodology. *Autonomous Agents and Multi-Agent Systems*, *8*(3), 203-236.

Casati, F., Sayal, M., & Shan, M. (2001). Developing e-services for composing eservices. In K. Dittrich, A. Geppert, and M. Norrie (Eds.), *Advanced Information Systems Engineering, 13th International Conference, CAiSE 2001, Interlaken, Switzerland, June 2001, Proceedings.* Springer.

Dalpiaz, F., Paja, E., & Giorgini, P., (2011). Security Requirements Engineering via Commitments. To appear in *First Workshop on Socio-Technical Aspects in Security and Trust (STAST'11).*

Devanbu, P.T., & Stubblebine, S. (2000). Software engineering for security: a roadmap. In A. Finkelstein (Ed.), *Proceedings of the Conference on the Future of Software Engineering* (pp. 227-239). Limerick, Ireland: ACM.

Dillon, A. (2001). Evaluation of software usability. In W. Karwowski (Ed.), *Encyclopedia of Human Factors and Ergonomics* (pp. 1110-1112). London: Taylor and Francis.

Dix, A. J., Finlay, E., Abowd, G. D., & Beale, R. (2003). *Human-Computer Interaction (3rd Edition)*. NJ, USA: Prentice-Hall.

Giorgini, P., Massacci, F., & Mylopoulos, J. (2003). Requirement Engineering meets Security: A Case Study on Modeling Secure Electronic Transactions by VISA and Mastercard. In I. Song, S. W. Liddle, T. W. Ling, and P. Scheuermann (Eds.), *Conceptual Modeling – ER 2003, 22nd International Conference on Conceptual Modeling, Chicago, IL, USA, October 13-16, 2003, Proceedings* (pp. 263-276). Springer.

Giorgini, P., Massacci, F., Mylopoulos, J., & Zannone, N. (2005). Modeling Security Requirements through Ownership, Permission and Delegation. In *13th IEEE International Conference on Requirements Engineering RE 2005 29 August - 2 September, 2005, Paris, France* (pp. 167-176). Los Alamitos: IEEE Computer Society.

Golafshani, N. (2003). Understanding Reliability and Validity in Qualitative Research. *The Qualitative Report*, *8*(4), 597-606.

Hix, D., & Hartson, H. R. (1992). *Formative Evaluation: Ensuring Usability in User Interfaces*. Technical Report, Virginia Polytechnic Institute & State University, Blacksburg, VA, USA.

Hommes, B. J., & van Reijswoud, V. (2000). Assessing the quality of business process modeling techniques. In *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences* (pp. 4-7). IEEE.

International Standard: ISO 9241-11:1998: *Ergonomic requirements for office work with visual display terminals (VDTs) – Part 11: Guidance on usability.* Retrieved December 8, 2011, from http://www.iso.org/iso/catalogue_detail.htm?csnumber=16883

International Standard: ISO 9241-210:2010: *Ergonomics of human-system interaction – Part 210: Human-centred design for interactive systems.* Retrieved December 8, 2011, from http://www.iso.org/iso/catalogue_detail.htm?csnumber=52075

Jürjens, J. (2002). UMLsec: Extending UML for Secure Systems Development. In J. Jézéquel, H. Heinrich, and S. Cook (Eds.), *«UML» 2002 - The Unified Modeling Language* (pp. 1-9). Berlin/Heidelberg: Springer.

Kahlaoui, A., Abran, A., & Lefebvre, E. (2008). DSML Success Factors and Their Assessment Criteria. In A. Abran, M. Bundschuh, R. Dumke, C. Ebert, and H. Zuse (Eds.), *Software Measurement News*, *13*(1), 43-51.

Kamandi, A., & Habibi, J. (2008). Modeling Languages Study and Evaluation Techniques. In D. Al-Dabass, S. Turner, G. Tan, and A. Abraham (Eds.), *AICMS 2008. Second Asia International Conference on Modeling & Simulation* (pp.553-558). Los Alamitos, California: IEEE.

Kärnä, J., Tolvanen, J. P., & Kelly, S. (2009). Evaluating the Use of Domain-Specific Modeling in Practice. In M. Rossi, J. Sprinkle, J. Gray, and J. Tolvanen (Eds.), *Proceedings of the 9th OOPSLA Workshop on Domain-Specific Modeling (DSM'09*) (pp. 14-20). HSE Print.

Massacci, F., Mylopoulos, J., & Zannone, N. (2010). Security Requirements Engineering: The SI* Modeling Language and the Secure Tropos Methodology. In Z. Ras, and L. Tsay. (Eds.), *Advances in Intelligent Information Systems* (pp. 147-174). Berlin/Heidelberg: Springer.

McDermott, J., & Fox, C. (1999). Using Abuse Case Models for Security Requirements Analysis. In *Proceedings of the 15th Annual Computer Security Applications Conference (ACSAC'99)* (pp. 55-64). IEEE.

Moody, D., & van Hillegersberg, J. (2009). Evaluating the Visual Syntax of UML: An Analysis of the Cognitive Effectiveness of the UML Family of Diagrams. In D. Gaševic, R. Lämmel, and E. Van Wyk (Eds.), *Software Language Engineering* (pp. 16-34). Berlin/Heidelberg: Springer.

Nysetvold, A., & Krogstie, J. (2005). Assessing Business Processing Modeling Languages Using a Generic Quality Framework. In T.A. Halpin, K. Siau, and J. Krogstie (Eds.), *Proceedings of the CAiSE'05 Workshops* (pp. 545-556). Idea Group.

Opdahl, A. L., & Sindre, G. (2009). Experimental comparison of attack trees and misuse cases for security threat identification. *Information and Software Technology, 51*, 916–932.

Paige, R. F., Ostroff, J. S., & Brooke, P. J. (2000). Principles for modeling language design. *Information and Software Technology, 42*(10), 665-675.

Recker, J. C., zur Muehlen, M., Siau, K., Erickson, J., & Indulska, M. (2009). Measuring method complexity: UML versus BPMN. *AMCIS 2009 Proceedings*. Paper 541. Retrieved December 8, 2011, from http://aisel.aisnet.org/amcis2009/541

Rosson, M. B., & Carroll, J. M. (2001). *Usability Engineering: Scenario-Based Development of Human-Computer Interaction*. San Francisco: Morgan Kaufmann Publishers Inc.

Scriven, M. (1967). The Methodology of Evaluation. In R. Tyler, R. Gagne and M. Scriven (Eds.), *Perspectives of Curriculum Evaluation* (pp. 39-83). Chicago: Rand McNally.

Sindre, G., & Opdahl, A. L. (2005). Eliciting Security Requirements with Misuse Cases. *Requirements Engineering, 10*(1), 34-44.

Wilson, C. E. (2006). Triangulation: the explicit use of multiple methods, measures, and approaches for determining core issues in product development. *Interactions, 13*(6), 46ff.

---

[i] Aniketos (www.aniketos.eu) is about ensuring trustworthiness and security in composite services
[ii] http://docs.oasis-open.org/wsbpel/2.0/CS01/wsbpel-v2.0-CS01.html
[iii] http://www.eclipse.org/community/rcp.php
[iv] http://www.eclipse.org/modeling/gmp/