2011 Sixth International Conference on Availability, Reliability and Security

Experimental Comparison of Misuse Case Maps with Misuse Cases and System Architecture Diagrams for Eliciting Security Vulnerabilities and Mitigations

Peter Karpati[(1)] Andreas L. Opdahl[(2)] Guttorm Sindre[(1)]

(1) Dept. of Computer and Information Science
Norwegian University of Science and Technology
Trondheim, Norway
{kpeter,guttors}@idi.ntnu.no

(2) Dept. of Information Science and Media Studies
University of Bergen
Bergen, Norway
Andreas.Opdahl@uib.no

***Abstract*—The idea of security aware system development from the start of the engineering process is generally accepted nowadays and is becoming applied in practice. Many recent initiatives support this idea with special focus on security requirements elicitation. However, there are so far no techniques that provide integrated overviews of security threats and system architecture. One way to achieve this is by combining misuse cases with use case maps into misuse case maps (MUCM). This paper presents an experimental evaluation of MUCM diagrams focusing on identification of vulnerabilities and mitigations. The controlled experiment with 33 IT students included a complex hacker intrusion from the literature, illustrated either with MUCM or with alternative diagrams. The results suggest that participants using MUCM found significantly more mitigations than participants using regular misuse cases combined with system architecture diagrams.**

Keywords: security requirements, intrusion analysis, architectural view, misuse cases, misuse case maps, controlled experiment

I. INTRODUCTION

In recent years, a variety of methods and techniques have been proposed for dealing with security requirements in software development projects. The Common Criteria method [1] provides, among other things, a classification of various types of functional security requirements. A sophisticated taxonomy for security requirements is provided in [2]. A number of diagrammatic modelling techniques have been suggested, such as attack trees [3], threat trees [4] and variations [41-45], in addition to security adaptations of modelling approaches developed for software system development in general [5-19], to be reviewed in Section III.

Despite the many techniques and methods available, existing modelling approaches focus on either user-level security issues of system-level security architecture. We have found none that integrate overviews of user-oriented security threats and mitigations with system architecture. Yet security threats and architecture considerations are related. During early RE, elicitation and analysis of security requirements take place in an architectural context that is central for understanding the potential security vulnerabilities and possible mitigations for the new system. In the transition from RE to architecture design, security requirements, along with other quality requirements, play an important role when selecting among the alternative architectures.

This paper contributes to filling the gap between security and architecture by investigating a new technique called *misuse case maps* (MUCM), which attempts to combine the most useful features of misuse cases [8] and of use case maps [20,21]. Our research question is *"Will a combined diagram notation for security attacks and system architecture (MUCM) be better for finding vulnerabilities and mitigations than using separate diagram notations?*"

This question is addressed through a controlled experiment focusing on vulnerabilities and mitigations. The next two sections present the *Background* for our research and *Related Work*. We then present our *Research Method* and *Results*, followed by a *Discussion.* Finally, we present a *Conclusion and Further Work.*

II. BACKGROUND

1. Misuse cases (MUC)

Misuse cases (MUC) [8] have become popular for security requirements elicitation and threat modelling. They complement use cases (UC) for security purposes by extending them with *misusers*, *misuse cases* and *mitigation use cases*, as well as new relations like *threatens* and *mitigates*. MUC diagrams use an inverted notation and are combined with regular use case diagrams. They represent security issues from the view of an attacker.

MUCs allow an early focus on security in the development process and facilitate discussion among stakeholders including regular developers with no special security training. However, they are not equally fit for all cases, for example they do not provide an integrated view of attacks and of system architecture.

2. Use case maps (UCM)

The use case map (UCM) notation was introduced in 1992 by Buhr and his team at Carleton University [20,21] and quickly gained popularity. Several papers were published in relation with UCM covering a wide range of fields. Today, UCMs are used in both research and industry.

A UCM presents an outline of the system's architecture and its relation to chosen behaviour scenarios, thus allowing a high-level understanding for design and development [21,22]. UCMs facilitate the discovery of problems among

978-0-7695-4485-4/11 $26.00 © 2011 IEEE
DOI 10.1109/ARES.2011.77

the collected use cases (UC) They can also be used to synchronize the UCs to check them for completeness, correctness, consistency and ambiguity and to identify differences in their abstraction levels. UCMs are also useful for documenting systems.

The basic UCM notation consists of three types of building blocks: 1) *runtime components* as rectangular boxes, 2) *responsibilities* attached to components as crosses and 3) *scenario paths* capturing a causal sequence of responsibilities (wiggly lines cutting through the responsibilities). A scenario path starts with a filled circle representing pre-conditions or triggering causes and ends in a bar representing post-condition or resulting effect. A UCM can include multiple scenarios to represent a chosen aspect of the system's behaviour.

3. *Misuse case maps (MUCM)*

Misuse case maps (MUCMs) [23] combine misuse cases (MUC) and use case maps (UCM), presenting an integrated view of security issues and system architecture. MUCMs address security requirements by focusing on vulnerabilities, threats and intrusions (inherited from MUCs) from an architectural point of view (inherited from UCMs). Their notation is based on the UCM notation and extended by vulnerabilities, exploits and mitigations. [23] presents a preliminary evaluation of MUCM with 12 participants as proof of concept though a more thorough investigation was needed which is described in the following sections.

Fig. 1 shows the most relevant part of MUCM notation. It extends UCM's basic notation with *intrusions*, represented by one or more *exploit paths* that cut through vulnerable parts of the system. Each path starts with a triangle. If no damage happens, it ends in a bar, just like in UCMs. Otherwise, the exploit path ends in a lightning symbol. Exploit paths can be numbered as individual steps in a complex intrusion. The step of an intrusion will mostly be causally related, each building on the results of the previous ones. Responsibilities can be left out if they are not relevant for the overall intrusion. Fig. 2 depicts an example showing an intrusion into a banking computer system and is summarized in the next paragraph (source [24]).

First, the intruder found an interesting bank by browsing a web site with organizations and IP ranges assigned. Next, he probed for further details about the IP addresses of the bank and found a server that was running Citrix MetaFrame (remote access software). He then scanned other networked computers for the remote access port to Citrix terminal services (port 1494). The attacker knew he might be able to enter the server with no password, as the default Citrix setting is "no password required". He searched files on the computer for the word "password" to find the clear text password for the bank's firewall and then tried to connect to routers and found one with default password. He added a firewall rule allowing incoming connections to port 1723 (VPN). After successfully authenticating to the VPN service, the attacker's computer was assigned an IP address on the internal network, which was flat, with all systems on a single segment. He discovered a confidential report written by security consultants containing a list of network vulnerabilities. In this case, the attacker was a white hat hacker and claimed not to cause any harm nor misuse any information.

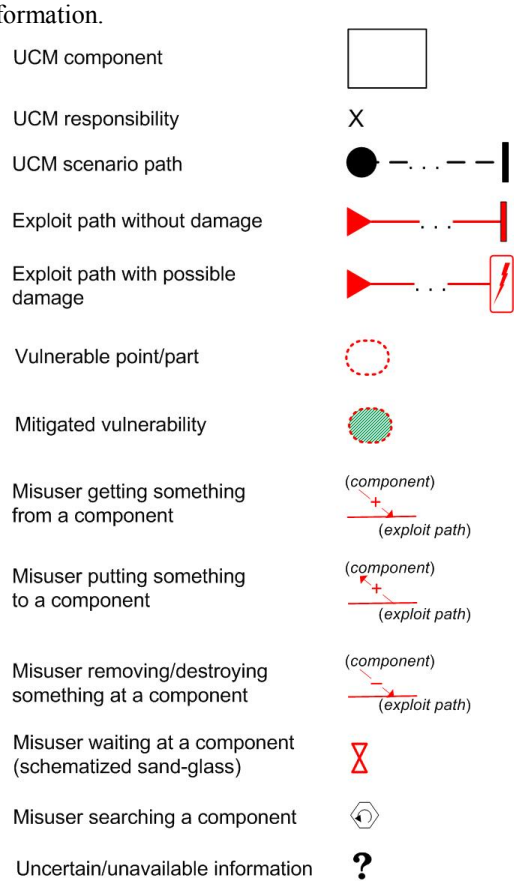

Figure 1. The MUCM notation.

The system may have vulnerable points (such as authentication responsibility) or parts (such as components without up-to-date security patches) which are suspect to threats. Mitigations can help to counter the threats and appear in the MUCM as desired possibilities (e.g. change default settings when adequate) which translate to security requirements. Individual cases of misuse are depicted by an exploit path crossing a vulnerable point or the boundary of a vulnerable part. The notation is rich and offers many other possibilities that we will not present here.

III. Related Work

Misuse Case Maps (MUCM) is not the only attempt to make a security adaptation of a more mainstream modelling technique. Attack trees [3] place a high-level attack in the root node of the tree and decompose it through AND/OR branches into lower-level attacks that must succeed to realize the higher-level ones. They are intuitive models and many extensions of them appeared such as defense trees [41], protection trees [42], attack response trees [43], attack countermeasure trees [44] and unified parametrizable attack trees [45]. Secure i* [5] is an extension of the i* modelling

language, where malicious actors and their goals are modelled with inverted variants of the usual icons. Abuse frames [6] extend problem frames with anti-requirements that might be held by potential attackers. Abuse cases [7], misuse cases [8], and security use cases [9] are security-oriented variants of regular use cases. Secure Tropos [10] extends the Tropos method with security-related concepts for ownership, permission and delegation, whereas KAOS has been extended with anti-goals [11], which can be used to express the goals of an intruder who aims to exploit or otherwise harm the system. The CORAS project [12] combined misuse cases with UML-based techniques into a comprehensive method for secure systems development. Languages for secure business process modelling have been proposed based on BPMN [13] and UML activity diagrams [14]. Variants of UML for secure design include UMLsec [15] and SecureUML [16]. Security patterns describe recommended designs for security [17]. The formal specification language Z has also been used to specify security-critical systems [18, 19]. However, none of these other techniques provide an integrated diagram view of security threats and architecture.

[25] discusses security in software architectures by examining how composition of applications affects security and proposing a protection model where the security policy and the application code are separately programmed. This is a well defined suggestion conceptually, but it does not offer any visual representation of the models. [26] and [27] each propose a systematic approach for modelling and analysing security system architectures based on the Software Architecture Model (SAM) [28]. [26] bases the secure design on patterns derived from security policies, and therefore lacks the notion of threat or vulnerability. [27] supports the concept of threats to assets and identifies and categorizes access points using threat categories from STRIDE [29], but it has no diagram notation for these threats. STRIDE is also used in [30, 31], but then in connection with misuse cases (MUC) to aid the definition of a secure architecture. Another related approach is misuse sequence diagrams, which was experimentally compared with MUCM in [32]. Misuse sequence diagrams may partially indicate the system architecture through objects and lifelines, but unlike MUCM they do not give an overview of the composition of components. [33] proposes a different type of architecture representation to separate threats from architecture. The approach uses a model-driven approach to architect secure software, defining the core elements of software architecture similarly to [25] and mapping them to a light-weight extension of the UML meta model. UML component diagrams are used to represent the architectural model to establish traceability of security objectives from the requirements to the architecture and support design-time security analysis. This approach starts with established security requirements and policies and therefore does not need the modelling of vulnerabilities and threats.

IV. RESEARCH METHOD

To investigate whether the MUCM notation was better than a traditional approach that shows attacks and architecture in separate diagrams, we ran a controlled experiment where both notations were used to perform the same tasks. A crucial question then is what to choose as control notation for the comparison. Except for the difference of showing attacks and architecture in one diagram (MUCM, cf. fig 2) or two (the *control notation*, or *null treatment*), the two alternatives should resemble each other as closely as possible, to avoid confounding factors in the experiment. Hence, the other alternative was chosen to be misuse case diagrams plus system architecture diagrams (henceforth abbreviated: MUC&SAD), with the SAD displaying architecture in exactly the same way as the MUCM, except that exploitation paths were removed. There are of course many other possible ways of displaying system architecture but, as shown in the Related Work section, none of them would be a perfect fit candidate for an alternative notation to use together with MUCM in the experiment. Although there are possibilities such as [25, 27, 30, 31], none of them can be considered established, obvious null treatments. Choosing one of them would also have made the comparison more complicated and would not have fit the research question equally well. We will address these candidates for further comparisons in the Discussion section.

A. Research approach

A controlled experiment was set up, with 33 subjects divided into two groups. To test problem solving effectiveness, they were then asked to identify as many vulnerabilities and mitigations as possible for the system, since these are commonly recommended security requirement tasks [34]. Finally, they were given a post-task questionnaire with 12 questions inspired by the Technology Acceptance Model (TAM, [35]), rating Perceived Ease of Use (PEOU), Perceived Usefulness (PU), and Intention to Use (ITU) on a 5-point Likert scale.

B. Hypotheses

The individual hypotheses for our study were:

$H1_1$: The MUCM group will find more vulnerabilities than the control group.

$H2_1$: The MUCM group will find more mitigations than the control group.

$H3_1$: The MUCM group will find more vulnerabilities and mitigations than the control group.

$H4_1$: The MUCM group will perceive their technique more favourably than the control group.

In addition to the hypotheses, we also anticipated a detailed qualitative and quantitative analysis on the types of vulnerabilities and mitigations identified.

C. Experimental design and procedure

The 33 participants were recruited from a class of 3rd year IT students, whose excursion organization received some payment for each participant. The group using MUCM consisted of 17 people, whereas the MUC&SAD group had

16. Both groups solved the task under equal conditions: same room, same time-frame, going through the following steps:

1. Filling in the pre-experiment questionnaire
2. Reading a short introduction to the experiment
3. Reading the introduction to the techniques
4. Reading the textual description of the bank hack and looking at the appropriate diagrams
5. Solving the tasks
6. Answering the post-task questionnaire

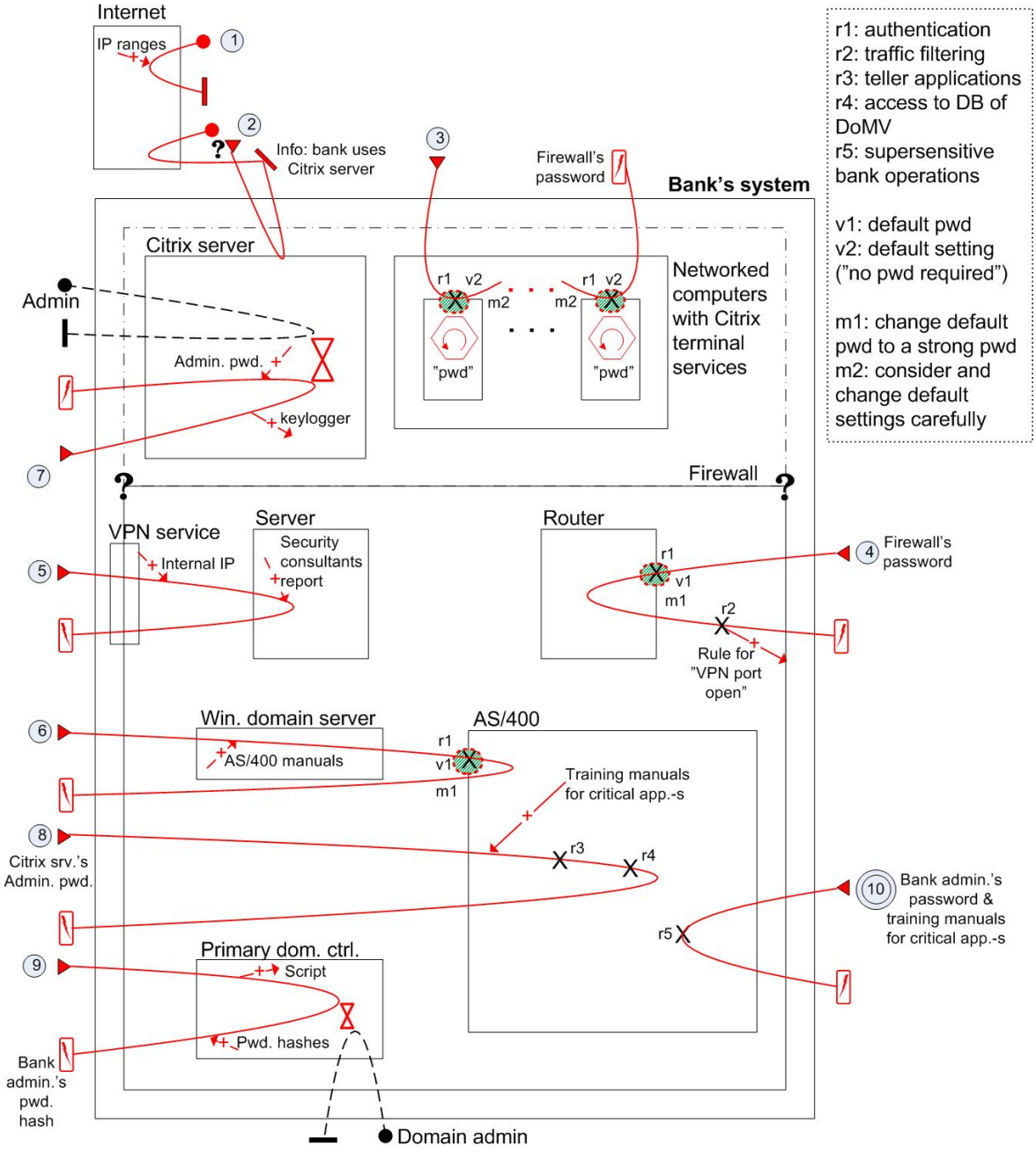

Figure 2. MUCM for the bank hack example from the experiment.

V. RESULTS

A. Comparing backgrounds

The participants were asked about their experience with system modelling, misuse cases, use case maps, misuse case maps and security analysis on a 5-point Likert scale. They also had to fill in their numbers of completed semesters of IT studies and their months of relevant work experience.

The data were not normally distributed so, here and later in the Results, we used non-parametric Mann-Whitney U tests with exact significances and 0.95 confidence levels. We

found no significant differences in the groups' backgrounds, except for the months of work experience benefiting the MUCM group (*p = 0.027, two-tailed*). We carefully controlled for the effect of this difference throughout the analysis, as we will explain in the *Discussion* section.

A. Coding

Whereas the questionnaires and responses to true/false questions lent themselves directly to statistical analysis, the answers to the tasks of identifying vulnerabilities and mitigations had to be coded in order to determine exactly how many vulnerabilities and mitigations each participant had identified in their responses. Coding was supported by NVivo9 and spreadsheet software and carried out in parallel by the two first authors, who went through several coding iterations before their codings converged. In addition to counting the vulnerabilities and mitigations, they were also categorised using multi-layered typologies of vulnerabilities and mitigations that emerged as part of the coding process. The purpose was to quantify how many relevant identifications each participant had made within each category. In each coding iteration, each identified vulnerability was assigned by each coder to its most appropriate type. For example, when one experiment subject had written "Storing passwords in the system", this could be coded by one coder in one iteration as a *Vulnerability* in the main category *Bad passwords* and in the sub-category *Weak password management*. At the end of each iteration, the two coders compared and revised their typologies and attempted to align their coding practices. After several iterations, the hierarchies and the coding had converged so that the remaining differences could be settled by phone and email. Whereas the early iterations had used hierarchies of depth three and four, the final vulnerability tree had two levels, with eight vulnerability types on the top level and fifteen subtypes on the level below. The mitigation tree had only one level with eight mitigation types that matched the eight vulnerability types.

The coding had revealed that one subject in the MUCM had produced almost no valid vulnerabilities and mitigations. This participant was removed from the dataset as an outlier. Between them, the remaining 32 participants had identified 242 vulnerabilities and 214 mitigations.

B. Analysing the performance

After coding, the performance of the two treatment groups could be compared. We conducted Mann-Whitney U tests both for the sum of vulnerabilities and mitigations, for the total numbers of vulnerabilities and mitigations and for their various subtypes. The results are summarized in Table I. The MUCM group did better than the MUC&SAD group both for vulnerabilities and mitigations, but the difference was only significant for mitigations. The pooled standard deviations for were 2.65 for vulnerabilities, 4.52 for mitigations and 6.21 for their sum, resulting in respective effect sizes (Cohen's d) of 0.33, 0.66 and 0.72. The latter two are classified by Cohen [36] and Hopkins [37] as moderate while the former effect is only small.

There were no significant differences between the two groups for any of the vulnerability subtypes. For the mitigation subtypes, there were significant differences for two types: *Improve network structure* (*p=0.022, one-tailed*) and *Improve passwords* (*p=0.016, one-tailed*). Further studies are needed to investigate whether MUCMs are particularly effective for these two types of mitigations. Although there were few significant differences on the level of particular vulnerabilities and mitigations, the tendency was that MUCMs were more effective. For vulnerabilities, the MUCM group found more of every main type, except for *Unprotected documents* and *Weak network architecture*. For mitigations, the MUCM group found more of every type. It is a bit surprising that the MUCM group found fewer vulnerabilities relating to *Weak network architecture*, but significantly more mitigations of the corresponding *Improve network structure* type. It reflects that numbers of vulnerabilities and mitigations were not strongly correlated (*Spearman's rho = 0.259*) for the pooled sample so our most detailed results must be used with caution.

In order to investigate why the relative performance of MUCMs was better for mitigations than for vulnerabilities, we conducted further correlation analyses using Spearman's rho. Although the numbers of vulnerabilities and of mitigations were not significantly correlated for the pooled sample, the correlation became significant for the MUCM group *(rho = 0.574, p = 0.020)*, but not for the MUC&SAD group *(rho = -0.019)*, when analysing the two groups separately. This pattern repeated itself for correlation between matching pairs of vulnerability and mitigation subtypes, such as vulnerability subtype "Weak software" and mitigation subtype "Protect software". For the MUCM group, most of the subtypes were correlated and some of them strongly. For the MUC&SAD group, most were not correlated. This offers an explanation of why the MUCM group performed significantly better only for mitigations. It is possible that, if used for identifying only vulnerabilities or only mitigations, MUCM would be slightly better than MUC&SAD for both purposes, but not significantly so. But when the techniques were used *first* for identifying vulnerabilities and *then* for mitigations, MUCM had the additional advantage of better supporting identification of mitigations based on the previously identified vulnerabilities. It was possibly this additional advantage that made the difference between MUCM and MUC&SAD significant for mitigations in the experiment (but it could not help for vulnerabilities, which had already been identified at this stage in the experiment).

In summary, hypothesis $H1_1$ must be rejected, instead accepting the null hypothesis of no difference between the techniques. On the other hand, hypotheses $H2_1$ and $H3_1$ can be accepted.

C. Analysing perceptions

While the participants performed better with MUCM, the data from the post-task questionnaires indicated a slight preference for MUC&SAD though far from significant. So, hypothesis $H4_1$ must be rejected, instead accepting the null hypothesis of no difference between the techniques.

TABLE I. NUMBER OF VULNERABILITIES AND MITIGATIONS FOUND IN THE EXPERIMENT

	MUCM	MUC&SAD	MUCM	MUC&SAD	MUCM	MUC&SAD	W	Exact sig., one-tailed
Vulnerabilities	16	16	8.00	7.13	17.56	15.44	247.0	(p = 0.264)
Mitigations	16	16	8.50	4.88	20	13	208.0	p = 0.017
Sum vuln+mitig.	16	16	16.50	12.00	19.97	13.03	208.5	p = 0.018

VI. DISCUSSION

A. Main findings

Our main findings are that MUCM performed better than MUC+SAD, in particular for identifying mitigations. There are also interesting differences when we consider particular subtypes of vulnerabilities and mitigations, but we have so far found few significant results. The significant correlations between vulnerabilities and mitigations identified by the MUCM group offers a possible explanation why the differences were significant only vulnerabilities. The difference in performance was not matched by any similar difference in preference for the techniques. However, it has also been observed in similar experiments previously that perceptions of a technique and performance with the same technique are not necessarily correlated, e.g. [38].

B. Implications for the technique

The MUCM notation is a recent research direction with many improvement opportunities remaining. Although the alternative technique combination was not significantly better perceived, the findings about perception are corroborated by comments we received in the preliminary evaluation of [23] regarding the complexity of the notation and of the map of an elaborate intrusion. Before starting to think about how to balance between the expressiveness and the readability of the notation, we have to know the target group. We believe that MUCMs have the potential to be applied by stakeholders with technical modelling expertise. The notation is flexible enough to convey details in the necessary amount and serve this way as a light-weight or a heavy-weight tool depending on the usage circumstances.

The MUCM group also found more vulnerabilities than the control group but not significantly so. Although we have already suggested an explanation, this could depend on the setting as well since the experiment task was a penetration test aiming to collect security requirements for an existing system, not starting a new design from scratch.

C. Threats to validity

This section discusses the validity of our results, using the four categories defined in [39], i.e., conclusion, internal, construct and external validity.

Conclusion validity concerns the relationship between the technique used and the outcome in terms of scores for tests and post-task questionnaire. One important question is whether the sample size is big enough to justify the conclusions drawn in the significant cases. The main effect claimed from the performance data was a significant advantage for MUCM regarding number of mitigations found and total number of vulnerabilities and mitigations. Denoting the Type I error probability by α and the Type II error probability by β, the following relationship holds:

$$N = \frac{4(u_{\alpha/2} + u_{\beta})^2}{ES^2}$$

If we use α = 0.05 and β = 0.20, we get $N = 32/(ES)^2$ [37] as a required sample size or $ES = \sqrt{32/N}$ as required effect size. As shown in Table II, we do not have sufficient observations. Moreover, these effect sizes must be used with caution since our data are not normally distributed.

TABLE II. EFFECT SAMPLE SIZES FOR THE SIGNIFICANT RESULTS

	Effect size	Required sample size	Required effect size
1) Vulnerabilities	0.66	74	1.0
2) Vuln.-s and mitig.-s	0.72	62	1.0

Another aspect of conclusion validity is the risk of missing a relationship that was really present due to a too small sample size. For example, MUCM did not give significant results for the number of vulnerabilities, although they had a higher average than MUC&SAD. The effect size was 0.33 in this case, meaning that, if this difference was indeed a real effect of the treatment, significant results could have been obtained with 293 participants. However, this is a big sample size and the observed effect is so small that it is also reasonable to assume that the difference was due to chance. We can similarly consider the overall TAM questionnaire response (with required sample size 127) which indicates accepting the null hypothesis is reasonable.

Internal validity assesses whether the observed outcomes were due to the treatment or to other factors. To avoid the threat of selection bias, participants were randomly assigned to the two groups, and we also used a pre-experiment questionnaire to control for confounding factors. Furthermore, both groups were sitting in the same auditorium subject to the same time limits and other conditions. The auditorium was sufficiently big for the participants to sit far apart, and there was no interaction between them during the experiment, hence ensuring participant independence.

As mentioned in the analysis, the participants in the MUCM group had significantly longer work experience than the other group. This was caused in part by two MUCM-subjects with 20 and 36 job months respectively. We controlled for the difference in several ways. We repeated all the Mann-Whitney tests for hypotheses H1-H4 with the two outliers removed. All the tendencies remained, and hypotheses H2 (p = 0.44) and H3 (p = 0.41) could still be

confirmed We also repeated all the correlation analyses with the two outliers removed, confirming that all the main tendencies and significances were unchanged, and we conducted partial correlation analyses controlling for job months and its log transformation with the same results.

A final threat to internal validity is our choice of null treatment, i.e., the combined use of MUC and SAD diagrams. Although we consider this null treatment the best available, there are other options for further work. The proposal in [27] has a visual model of the architecture and should be considered for future comparisons, although it has no diagram notation for threats. The modelling technique in [30, 31] could also have been an appropriate comparison alternative, but there are two problems. The first one is the necessity of an extension of the software architecture concept used by the authors to a more general system architecture concept which can include all kinds of components possible targeted. The second problem is that the publicly available intrusion descriptions do not provide enough details on the components and their interactions to create an appropriate architecture diagram. UML component diagrams as used in [33] and deployment diagrams as used by UMLSec [15] are also possible future candidates for a comparison with MUCMs, although these UML diagrams do not model vulnerabilities and threats explicitly.

Construct validity concerns whether it is legitimate to infer from the measures made in the experiment to the theoretical constructs that one was trying to observe. With respect to the performance data, we were trying to observe the problem solving effectiveness achieved using the respective techniques, and this was measured in terms of the number of vulnerabilities and mitigations identified. Of course, there are other possible ways to explore effectiveness, but the identification of vulnerabilities and mitigations would be relevant problem solving tasks in secure software engineering.

With respect to the post-task questionnaire, we used 12 questions inspired by TAM. We were unable to demonstrate construct validity for this questionnaire instrument, as the results of factor analysis did not correspond to the constructed items PEOU, PU, and ITU. The sample size is also too small for factor analysis (recommendations typically say 100 or more). However, our purpose in this research was not to validate TAM but just to compare the participants' preference for the two techniques.

External validity asks whether it is possible to generalize from the experimental setting to other situations, most importantly to industrial systems development. The use of students instead of practitioners is a notable threat here, but our participants were soon to finish their third year of computing studies, thus not fundamentally different from people who enter the IT workforce with a 3-year Bachelor. Moreover, the task in this experiment was to learn a new method and employ it on a task explained during the experiment – the difference between students and practitioners would probably have been bigger if the task was about using a method well-known to the practitioners.

Another threat is limited realism of the experimental task. A controlled experiment can hardly last for more than a couple of hours, meaning that the case description and tasks that the participants are supposed to solve must be fairly simple, not representative of huge and complex problems that are faced in real development projects. However, we took care to use a realistic attack scenario written by an expert hacker, so our conclusions should at least be valid for understanding single scenarios, but cannot be generalized to understanding the security of entire systems. Such complex tasks, however, are hard to explore experimentally and must rather be investigated through case studies.

VI. Conclusions and Further Work

The paper has presented a first experimental evaluation of misuse case maps. The results indicate that misuse case maps is an improvement of misuse case and system architecture diagrams used in combination. Hence, our research question posed in the introduction can be answered positively: There are advantages of showing security issues and system architecture in the same diagram rather than in separate diagrams. In particular, there were significant differences when identifying mitigations. Misuse case maps also helped identifying more vulnerabilities but not significantly so. However, misuse case maps were perceived slightly less favourably than separate misuse cases and architecture diagrams, especially for questions related to ease of use and communication with non-experts. Although not significant, this tendency indicates that the advantages of showing more information in the same diagram also come at the cost of complexity, possibly reducing their user appeal.

Additional work is needed to explore misuse-case maps in several directions. Case studies can develop the technique further and clarify its relation to other requirements, architecture and security techniques and methods. A particular challenge is to find the right balance between expressiveness and ease of use and communication. Tool support will be needed to support larger and more realistic cases (an editor is under construction). Further experimental evaluations are also needed, e.g. to compare MUCM with other modelling approaches like [15,30,31,33]. Additional experiments with IT professionals are being prepared.

Acknowledgement

This work is part of the ReqSec project [40] funded by the Norwegian Research Council.

References

[1] CCIMB, Common Criteria for Information Technology Security Evaluation, Common Criteria Implementation Board, Technical Report, CCIMB-99-031, 1999.

[2] D.G. Firesmith, A Taxonomy of Security-Related Requirements, Proceedings of the International Workshop on High Assurance Systems (RHAS'05), Paris, France, 2005.

[3] B. Schneier, Secrets and Lies: Digital Security in a Networked World, Wiley, Indianapolis, 2000.

[4] E.J. Amoroso, Fundamentals of Computer Security Technology, Prentice-Hall, Upper Saddle River, NJ, 1994.

[5] L. Liu, E. Yu, J. Mylopoulos, Security and Privacy Requirements Analysis within a Social Setting, Proceedings of the 11th

International Requirements Engineering Conference (RE'03), IEEE Press, Monterey Bay, CA, 2003, pp. 151-160.

[6] L. Lin, B. Nuseibeh, D. Ince, M. Jackson, Using Abuse Frames to Bound the Scope of Security Problems, in: N.A.M. Maiden (ed.), Proceedings of the 12th IEEE International Requirements Engineering Conference (RE'04), IEEE, Kyoto, Japan, 2004.

[7] J. McDermott, C. Fox, Using Abuse Case Models for Security Requirements Analysis, Proceedings of the 15th Annual Computer Security Applications Conference (ACSAC'99), IEEE CS Press, 1999.

[8] G. Sindre, A.L. Opdahl, Eliciting Security Requirements with Misuse Cases, Requirements Engineering, 10 (2005), 34-44.

[9] D. Firesmith, Security Use Cases, Journal of Object Technology, 2 (2003), 53-64.

[10] P. Giorgini, F. Massacci, J. Mylopoulos, N. Zannone, Modelling Security Requirements Through Ownership, Permission, and Delegation, in: J. Atlee, and C. Rolland (eds.), Proceedings of the 13th International Requirements Engineering Conference, IEEE, Paris, 2005, pp. 167-176.

[11] A. van Lamsweerde, S. Brohez, R. De Landtsheer, D. Janssens, From System Goals to Intruder Anti-Goals: Attack Generation and Resolution for Security Requirements Engineering, In Proc. 2nd International Workshop on Requirements Engineering for High Assurance Systems (RHAS'03), 2003, pp. 49-56.

[12] T. Dimitrakos, B. Ritchie, D. Raptis, J.Ø. Aagedal, F.d. Braber, K. Stølen, S.H. Houmb, Integrating model-based security risk management into eBusiness systems development - the CORAS approach, In Proc. 2nd IFIP Conference on E-Commerce, E-Business, E-Government (I3E'2002), Kluwer, 2003, pp. 159-175.

[13] A. Rodriguez, E. Fernandez-Medina, M. Piattini, Towards an Integration of Security Requirements into Business Process Modeling, In Proc. 3rd International Workshop on Security in Information Systems (WOSIS), INSTICC Press, 2005, pp. 287-297.

[14] A. Rodriguez, E. Fernandez-Medina, M. Piattini, Capturing Security Requirements in Business Processes through a UML 2.0 Activity Diagrams Profile, in: J. F. Roddick et al. (eds.), Proceedings of the Advances in Conceptual Modeling - Theory and Practice, ER 2006 Workshops, Lecture Notes in Computer Science Vol. 4231, Springer, Tucson, AZ, USA, 2006.

[15] J. Jürjens, UMLsec: Extending UML for Secure Systems Development, in: J. M. Jezequel, H. Haussmann, and S. Cook (eds.), Proceedings of the The Unified Modeling Language, 5th International Conference (UML 2002), Lecture Notes in Computer Science Vol. 2460, Springer, Dresden, Germany, 2002, pp. 412-425.

[16] T. Lodderstedt, D. Basin, J. Doser, SecureUML: A UML-Based Modeling Language for Model-Driven Security, in: J.M. Jezequel, H. Haussmann, and S. Cook (eds.), Proceedings of the The Unified Modeling Language, 5th International Conference (UML 2002), Lecture Notes in Computer Science Vol. 2460, Springer, Dresden, Germany, 2002, pp. 426-441.

[17] M. Schumacher, E.B. Fernandez, D. Hybertson, F. Buschmann, P. Sommerlad (Eds.), Security Patterns: Integrating Security and Systems Engineering, J. Wiley & Sons, Chichester, UK 2005.

[18] A. Boswell, Specification and Validation of a Security Policy Model, IEEE Transactions on Software Engineering, 21 (1995), 63-68.

[19] A. Hall, R. Chapman, Correctness by construction: developing a commercial secure system, IEEE Software, 19 (2002), 18-25.

[20] D. Amyot: Use Case Maps Quick Tutorial Version 1.0, 1999

[21] R. J. A. Buhr, R. S. Casselman: Use Case Maps for Object-Oriented Systems, Prentice Hall, 1995

[22] R. J. A. Buhr: Use case maps for attributing behaviour to system architecture, Proceedings of the 4th International Workshop on Parallel and Distributed Real-Time Systems, p.3, 1996

[23] P. Karpati, G. Sindre, A. L. Opdahl: Illustrating Cyber Attacks with Misuse Case Maps, In proceedings of 16th International Working Conference on Requirements Engineering: Foundation for Software Quality (RefsQ 2010)

[24] K. D. Mitnick, W. L. Simon: The Art of Intrusion, Wiley Publishing Inc. 2006

[25] C. D. Jensen. Secure Software Architectures. In Proceedings of the 8th Nordic Workshop on Programming Environment Research, pages 239–246, Aug. 1998.

[26] Y. Deng, J. Wang, J.P. Tsai, K. Beznosov, "An Approach for Modeling and Analysis of Security System Architectures," IEEE Transactions on Knowledge and Data Engineering, vol. 15, no. 5, pp. 1099-1119, Sept/Oct, 2003.

[27] Y. Ali, S. El-Kassas, M. Mahmoud, "A Rigorous Methodology for Security Architecture Modeling and Verification", In Proc.Hawaii Int'l Conf. on System Sciences (HICSS) , pp. 1-10, 2009

[28] J. Wang, X. He, Y. Deng, "Introducing Software Architecture Specification and Analysis in SAM through an Example," Information and Software Technology, vol. 41, no. 7, pp. 451-467, 1999.

[29] Michael Howard and David LeBlanc, "Writing Secure Code," Microsoft Press, 2002.

[30] J. J. Pauli, D. Xu: "Misuse Case-Based Design and Analysis of Secure Software Architecture". In Proceedings of the Int. Conference on Information Technology: Coding and Computing (ITCC'05). IEEE CS Press, April 2005.

[31] D. Xu, J. J. Pauli, "Threat-Driven Design and Analysis of Secure Software Architectures," Journal of Information Assurance and Security (JIAS), vol. 1, no. 3, pp 171-180, June 2006.

[32] V. Katta, P. Karpati, A. L. Opdahl, G. Sindre, C. Raspotnig: "Comparing two misuse modelling techniques for intrusion visualization". In Proc. POEM' 2010, Springer LNBIP vol 60.

[33] E. A. Oladimeji, S. Supakkul, L. Chung: "A Model-driven Approach to Architecting Secure Software". In Proc. SEKE 2007

[34] I. A. Tøndel, M. G. Jaatun, P. H. Meland.: "Security Requirements for the Rest of Us: A Survey". IEEE Software. vol. 25, no. 1, pp. 20-27, 2008

[35] F. D. Davis, R. P. Bagozzi, P. R. Warshaw: User Acceptance of Computer Technology: A Comparison of Two Theoretical Models. Management Science, 35 (1989) 982-1003

[36] J. Cohen (1988). Statistical Power Analysis for the Behavioral Sciences (second ed.). Lawrence Erlbaum Associates.

[37] W. G. Hopkins: A New View of Statistics. University of Queensland, Australia, Brisbane (2009), http://www.sportsci.org/resource/stats/

[38] A. L. Opdahl, G. Sindre: "Experimental Comparison of Attack Trees and Misuse Cases for Security Threat Identification", Information and Software Technology, 51(5):916-932, 2009.

[39] C. Wohlin, P. Runeson, M. Höst, M.C. Ohlsson, B. Regnell, A. Wesslén, Experimentation in Software Engineering: An Introduction, Kluwer Academic, Norwell, MA, USA, 2000.

[40] http://www.idi.ntnu.no/~guttors/reqsec

[41] S. Bistarelli, F. Fioravanti, P. Peretti: Defense trees for economic evaluation of security investments, In proceedings of ARES 2006. pp., 20-22

[42] K.S.Edge, G.C. Dalton, R.A. Raines, R.F. Mills: Using Attack and Protection Trees to Analyze Threats and Defenses to Homeland Security, MILCOM 2006. IEEE , vol., no., pp.1-7, 23-25 Oct. 2006

[43] S. A. Zonouz, H. Khurana, W. H. Sanders, and T. M. Yardley. RRE: A Game-Theoretic Intrusion Response and Recovery Engine. In Proc. DSN, pages 439–448, 2009

[44] A. Roy, D. Kim, and K. Trivedi. Cyber security analysis using attack countermeasure trees. In Proc. of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research, pp. 1–4, 2010.

[45] J. Wang, J.N. Whitley, R. C.W. Phan and D.J. Parish. Unified Parametrizable Attack Tree, International Journal for Information Security Research (IJISR), Volume 1, Issue 1, pp.20-26, 2011