Investigating Security Threats in Architectural Context: Experimental Evaluations of Misuse Case Maps

Peter Karpati[a][1], Andreas L. Opdahl[b], Guttorm Sindre[a]

(a) Dept. of Computer and Information Science, Norwegian University of Science and Technology

Sem Sælands vei 7-9, NO-7491 Trondheim, Norway

Peter.Karpati@hrp.no, Guttorm.Sindre@idi.ntnu.no

(b) Dept. of Information Science and Media Studies, University of Bergen,

P.O.Box 7802, NO-5020 Bergen, Norway

Andreas.Opdahl@uib.no

Paper submitted to Journal of Systems and Software, 2014.06.27.
Revision 1 submitted 2015.01.09.
Revision 2 submitted 2015.02.13.

Corresponding author:

Andreas L Opdahl

Postal address: Dept. of Information Science and Media Studies, University of Bergen,

P.O.Box 7802, NO-5020 Bergen, Norway

Email: Andreas.Opdahl@uib.no

Phone: +47 – 5558 4140

[1] *Present address:* Institute for Energy Technology, P.O. Box 173, NO-1751 Halden, Norway.

Investigating Security Threats in Architectural Context: Experimental Evaluations of Misuse Case Maps

Abstract*—Many techniques have been proposed for eliciting software security requirements during the early requirements engineering phase. However, few techniques so far provide dedicated views of security issues in a software systems architecture context. This is a problem, because almost all requirements work today happens in a given architectural context, and understanding this architecture is vital for identifying security vulnerabilities and corresponding mitigations. Misuse case maps attempt to provide an integrated view of security and architecture by augmenting use case maps with misuse case concepts. This paper evaluates misuse case maps through two controlled experiments where 33 and 54 ICT students worked on complex real-life intrusions described in the literature. The students who used misuse case maps showed significantly better understanding of intrusions and better ability to suggest mitigations than students who used a combination of two existing techniques as an alternative treatment. Misuse case maps were also perceived more favourably overall than the alternative treatment, and participants reported using misuse case maps more when solving their tasks.*

Keywords: *computer security, security requirements engineering, intrusion analysis, use case maps, misuse cases, misuse case maps, experimental evaluation*

1. Introduction

Security should be addressed already at the earliest stages of software development [1], and a variety of methods and techniques have recently been proposed for secure software development (e.g., [2]–[5]), including security requirements (e.g., [6]–[10]) and secure architecture design (e.g., [11]–[14]). However, few authors have focussed on the relationship between security requirements and software systems architecture, and very few methods or techniques (e.g., [15], [16]) have been proposed that provide dedicated overviews of high-level security issues in a software systems architecture context. This is a problem, because almost all requirements work today happens in a wholly- or partially-predetermined architectural context [17], [18], which is given, for example, by an *existing system* to be modified or extended; by the *neighbouring systems* it needs to communicate with; and by prexisting architecture and infrastructure *standards*. Understanding this, often complex, existing architecture is vital when identifying security vulnerabilities and corresponding mitigations during early requirements engineering. Even in later requirements stages, it has become widely accepted that software requirements and architecture design are concurrent activities [18], [19], especially in complex problem domains [20] where security often needs extra attention.

In previous work [16], [21], we have therefore proposed *misuse case maps (MUCM)*, which combine ideas from our own previous work on *misuse cases (MUC)* [7], [22] with ideas from Amyot, Buhr and Casselman's *use case maps (UCM)* [23]–[25]. The purpose of MUCM is to encourage investigation of potential security vulnerabilities and corresponding mitigations in a systems architecture context. The technique is intended to help identifying vulnerabilities of many different types and to help suggesting mitigations of many different types for each vulnerability. This paper presents an experimental evaluation of MUCM through two controlled student experiments, which extend our previous work on MUCM. In [16], we have presented a first, formative evaluation of MUCM through a small survey of researcher colleagues – the present paper instead uses controlled experimentation that involves more participants, that uses more realistic tasks, and that measures more variables. In [21], we have compared MUCM experimentally with another experimental technique (MUSD) – the present paper instead compares MUCM with existing notations that are commonly used in practice. In [26], [27], we have presented limited analyses of an experiment – the present paper reports the first full analysis of this experiment, including all dependent variables. The analysis of a second, additional experiment is presented for the first time here, along with an overall analysis of both experiments.

The purpose of our evaluation is to investigate further *whether MUCM is potentially a useful addition to the arsenal of modelling techniques* that developers and security experts have at their disposal while collaborating to develop secure software. And if so, we would like to know more about *how it is potentially useful, and why*. Because we are not aware of directly comparable techniques out there, our aim is not in any way to show that MUCMs are objectively "better than" or "superior to" existing techniques. Indeed, we advocate a multi-perspective approach to analysing requirements in general, and security requirements in particular, to which we hope MUCMs can provide one new and complementary perspective alongside existing ones. The rest of the paper is organised into related work – including our previous work on Misuse Cases and Misuse Case Maps – (Section 2.), research methods (Section 3.), results (Section 4.), discussion (Section 5.), and conclusion (Section 6.).

2. Background and Related Work

A number of techniques have been proposed for security requirements engineering, as can be seen for instance in surveys like [28]–[32]. Also, the relationship between security requirements engineering techniques and risk analysis has been reviewed by [33], and secure systems development onwards from requirements to design has been reviewed by [34] and [35]. Our purpose in this section is therefore not to make a complete review of previous research in the area, only to cover the most important techniques for security requirements (section 2.1), related system development techniques (section 2.2), secure architecture design (section 2.3), and the relationship between security requirements and architecture (section 2.4), in order to provide a context for our own work. Finally, section 2.5 will provide necessary background material for the rest of the paper, including our previously published work on Misuse Case Maps [16], [21], [26], [27], in order to make it possible to understand the starting point for this paper without having to read the previous papers.

2.1 Techniques and methods for security requirements

Modelling techniques and methods that are specific to security requirements work include, abuse cases [6], misuse cases [7], and security use cases [8], which are security-oriented variants of use cases [36]. Abuse and misuse cases represent behaviours that potential attackers want to perform using the system, whereas security use cases represent countermeasures intended to avoid or repel these attacks. The difference between abuse and misuse cases is that the latter show use and misuse in the same picture, whereas abuse cases are drawn in separate diagrams (for more details about misuse cases, see Section 2.5). Secure i* [9] is an extension of the i* modelling language, where malicious actors and their goals are modelled with inverted variants of the usual icons. [37] proposes extensions to i* to support systematic security requirements elicitation and analysis centered around vulnerabilities, attackers, and the security impacts of attacks. [38] extends i*with Analytic Hierarchy Process (AHP) analysis of security trade-offs. Abuse frames [10] extend problem frames with anti-requirements that might be held by potential attackers. Languages for secure business process modelling have been proposed based on both BPMN [39] and UML activity diagrams [40], [41]. [21] adapts UML sequence diagrams to security analysis.

Generic security techniques can also be used to analyse software security requirements, such as threat trees [42] and attack trees [43]. The latter represents a high-level attack as the root node of a tree so that it can be decomposed through AND/OR branches into lower-level attacks that must succeed in particular combinations for the high-level one to succeed. Attack trees are intuitive, and many extensions have appeared, such as defense trees [44], protection trees [45], attack response trees [46], attack countermeasure trees [47], and unified parametric attack trees [48]. The formal specification language Z has also been used to specify security-critical systems [49], [50].

The Common Criteria method [51] provides, among other things, a classification of various types of functional security requirements. A sophisticated taxonomy for security requirements is also provided in [52]. [53] integrates and extends the conceptual frameworks behind i*/Tropos and Problem Frames to a security ontology, grounded in DOLCE [54], in order to provide clear definitions of security concepts. [29] and [31] review and compare broad selections of current security requirements engineering approaches. [56] uses ontology to aid security requirements specification. [55] reviews and compares risk identification techniques for safety and for security requirements, whereas [56] presents ISSRM, a reference model for information system security risk management. [57] aligns misuse cases with ISSRM [56], and [58] develops it further into a framework for security risk management, which is used to align mal-activity diagrams with risk management in [59]. [60] proposes a method for coordinated elicitation and analysis of safety and security requirements.

Other researchers have also investigated the effectiveness and user perception of security requirements techniques empirically. [61] compares CORAS and SREP in a controlled experiment with 28 master students, measuring effectiveness by the students' performance of some tasks and perception by the participant responses to a TAM-based questionnaire. [62] empirically compares of four techniques – CORAS, Problem Frames, Secure I*, and Secure Tropos – through a qualitative investigation relying on observations, recordings, interviews, and focus group discussions. Other empirical evaluations of security modelling techniques are reviewed in [63].

2.2 Techniques and methods for secure software development

Other security techniques and methods attempt to cover later software development phases in addition to requirements. Secure Tropos [5] extends the Tropos method [64] with security-related concepts for ownership, permission, and delegation. It is adapted in [65] for security risk management in the early phases of information systems development based on ISSRM [56]. In [66], KAOS is extended with anti-goals [3] that can be used to express the goals of an intruder who aims to exploit or otherwise harm the system. The CORAS project [4] combined misuse cases with UML-based techniques into a comprehensive method for secure systems development. [15] proposes RiskREP, which extends misuse case-based methods with ICT-architecture based risk assessment and countermeasure selection, linking countermeasures to both business goals and costs.

Other variants of UML for secure design include UMLsec [2], SecureUML [67], and UMLintr [68]. UMLSec uses deployment diagrams to represent the physical architecture of a system and the distribution of the software components on the set of execution units. It is a complex method with limited scope in modelling security issues. It is more precise than the security modelling techniques we have mentioned so far, but primarily targets the design phase [2]. SecureUML is an adaptation of UML for formalizing access control policies for the development of secure distributed systems [67]. UMLIntr is a UML profile for describing intrusion scenarios (defined as “abstract description of the steps that an intruder follows to gain illegitimate access to some protected resources” in [68]), to be used in Intrusion Detection Systems (IDS).

[69] proposes SecReq, a security requirements engineering method that combines Common Criteria with HeRA, a heuristic requirements editor, and UMLsec to make security engineering knowledge more available to non-experts. [70] combines Secure Tropos with UMLsec to create a structured method for secure software systems development. The resulting method is able to transform security requirements to secure designs. [71] proposes to bridge security requirements analysis and secure architectural design by matching security requirement analysis patterns with corresponding architectural security patterns tailored to solve classes of security subproblems. The approach is based on UMLsec and UML2.3 [72]. Security patterns describe recommended designs for security [73].

[74] proposes to address an intertwining of requirements and design by integrating goal-oriented i* models with use case maps [23], [24], so that i* contributes the design rationale and use case maps represents the design (for more details about use case maps, see Section 2.5). Although [74] does not address security, we assume the later security-oriented extension of i* by the same authors [9] can be used in a similar way. Nevertheless, this style of modeling would focus on the goals of the attackers, more than on the vulnerabilities, mitigations or attacks themselves.

2.3 Techniques and methods for secure software architecture

[11] 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 programmed separately. This is a well defined suggestion conceptually, but it does not offer any visual representation of the models. [12] and [13] propose a systematic approach for modelling and analysing security system architectures based on the Software Architecture Model (SAM) [14]. [12] bases the secure design on patterns derived from security policies, and therefore lacks explicit notions of threat, vulnerability, and mitigation. [13] supports the concept of threats to assets and identifies and categorizes access points using threat categories from STRIDE [75], but offers no corresponding diagram notation. STRIDE is also used in [76], [77] in connection with misuse cases (MUC) to aid the definition of a secure architecture. Misuse sequence diagrams [21] partially indicate the system architecture through objects and lifelines, but do not give an overview of the composition of components. [78] proposes a different type of architecture representation to separate threats from architecture. The authors use a model-driven approach to architect secure software, defining the core elements of software architecture similarly to [11] 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 takes established security requirements and policies as its starting point and therefore does not support modelling of vulnerabilities and mitigations.

2.4 Considering security requirements in a software systems architecture context

Previous sections have shown that there are many available methods and techniques, both for security requirements and secure software architectures in particular and for secure software development in general. But far less attention has so far been paid to the relationship between security requirements and secure software systems architectures. Although a few of the methods and techniques we have already discussed address this relationship, none of them focus on considering security threats, vulnerabilities, and mitigations in a software systems architecture context: UMLSec [2] focusses on detailed security architecture, but does not emphasise investigation of security threats, vulnerabilities, and mitigations. Security patterns [73] can embed architecture considerations, but do not offer an approach to investigating security threats, vulnerabilities, and mitigations in an architectural context. UMLIntr [68] has a similar focus to our proposal, but it is more fine-grained in the sense that an intrusion in UMLIntr corresponds approximately to an individual attack step in MUCMs (for more details about misuse case maps, see Section 2.5). Secure Tropos combined with UMLsec [70] supports the transition from goal-driven security requirements investigation to architecture design, but treats security requirements investigation and secure architecture design as separate, sequential steps. UMLsec supported by UML security patterns [71] uses security requirements for pattern selection, but does not focus on investigating the potential security threats. RiskREP [15] assesses security threats in an architecture context, but on an enterprise architecture level that is higher than the systems architecture perspective of MUCMs. Use case maps (UCM) [23], [24] have been developed exactly with the purpose of investigating high-level systems functions in an systems architecture context, but does not explicitly consider security. Misuse patterns [79], however, capture connections between vulnerabilities and architecture. A difference from misuse case maps is that misuse patterns are represented through a textual template (Name, Context, Problem, Solution, Consequences, Countermeasures and Forensics, Known Uses, Related Patterns), and the example graphical models employ UML class and sequence diagrams.

2.5 **Misuse case maps (MUCM)**

To support investigation of security threats, vulnerabilities, and mitigations in a systems architecture context, we have previously proposed misuse case maps (MUCM) [16]. MUCMs combines features from system architecture diagrams (SAD), use case maps (UCM) [23], [24], and misuse cases (MUC) [7], [22], which we will therefore review each of them first.

The bank hack case: We will illustrate the techniques using a real case from the literature. Table 1 shows a stepwise description of an intrusion into a banking computer system [80], chapter 7, as reported by the perpetrator, who claims to be white-hat hacker, not misusing the retrieved information and not otherwise harming the system. A simplified version of this bank intrusion, along with the diagrams we will present in this section, were also used in our experiments.

System architecture diagrams (SAD): A System Architecture Diagram (SAD) typically depicts architecture by drawing every software and/or hardware component in the system as a labelled box. Boxes are nested to show component-subcomponent relations. (Also, arrows between boxes can be used to show uni- or bi-directional communication, but we have not needed to do this in the experiments.) SAD is a simple, informal, and intuitive notation that is much used in practice, although it has never been formally defined in a seminal paper or book.

Figure 1 shows one variant of this notation using the *bank hack* as an example (the diagram was also used in our experiments). The diagram shows two top-level components: the *Internet* and the *Bank's system*. The latter in turn has several sub-components, such as a *Citrix server*, several *Networked computers*, a *Server*, a *Router*, and several other system, surrounded by a *Firewall*. Somewhat unconventionally, the *VPN service* component is drawn on the border of the *Firewall* component to show that it is a boundary component that connects the inside components of the *Firewall* to the outside. Also unconventionally, the diagram indicates two extents of the *Firewall*: a smaller one drawn as a whole line and a larger one drawn as a dotted line that extends the smaller extent, suggesting that the *Citrix server* and other *Networked computers* are only apparently inside the *Firewall*.

Use case maps (UCM): A Use Case Map (UCM) [23]–[25] shows one or several behavioural scenarios (aka use cases) as arrows drawn across a system architecture diagram, thus providing a high-level view of a system and its behaviour [23], [24]. The basic UCM notation has three main elements: *runtime components* are drawn as rectangular boxes; *responsibilities* allocated to components are drawn as crosses; and *scenario paths* are sequences of responsibilities drawn as wiggly lines from cross to cross across the component boxes. A scenario path starts with a filled circle that represent pre-conditions or triggering causes and ends in a filled bar that represents post-conditions or effects.

UCM has been developed to integrate requirements level use cases with system architectures, explicitly showing how the steps in the use case is performed by components in the proposed architecture. It is used in both research and industry today. UCMs can be useful for many types of complex systems. They facilitate the discovery of problems among the collected use cases (UC). UCMs can also be used to check collections of related UCs for completeness, correctness, consistency, and ambiguity and to identify differences in their abstraction levels. Finally, UCMs are useful for documenting systems.

Figure 2 shows a Use Case Map of the bank hack case from Table 1. The diagram builds on the SAD diagram of Figure 1 (for experimental purposes we have chosen a variant of SAD that is maximally similar to the architecture backbone of UCMs). It shows two use cases. In the first case, a regular *Admin* accesses the *Bank's system* from the outside, going through the *Firewall* to access the *Citrix server.* In the second case, a *Domain admin* also accesses the *Bank's system* from the outside through the *Firewall*, in this case to access the bank's *Primary domain controller*.

Misuse cases (MUC): A Misuse Case (MUC) [7], [22] diagram extends a use case (UC) diagram for security purposes by explicitly showing *misusers*, *misuse cases*, and *mitigation use cases* alongside regular *use cases* and their *users*, using inverted icons to distinguish between regular and misusing behaviours. MUC diagrams also introduce new relation types like *threatens* and *mitigates*. The technique aims to facilitate discussion about security already in the early stages of information systems development, involving stakeholders who do not necessarily have a background in security or in software development. The technique is used in both research and practice for security requirements elicitation, threat modelling, and risk analysis.

Figure 3 shows how a MUC diagram for the bank hack case. In addition to regular users such as *Regular teller* and *System administrator*, the diagram shows an (grahically inverted) misuser icon representing the *Hacker*, whose (graphically inverted) misuse cases represent, for example, *Gaining teller priveleges* threatens the regular use case *Teller activities*. The MUC diagram also shows show misuse cases can include one another, as when *Gaining access to the server* includes *Intruding into the bank's network.* Not shown in Figure 3 is how (non-inverted) *mitigation use cases* (or *security use cases*) can in turn be introduced to *mitigate* misuse cases.

Misuse case maps (MUCM): A Misuse Case Map (MUCM) [16], [21] provides an integrated view of security threats and system architecture by extending UCMs with *vulnerabilities* and *exploit paths*. A *vulnerable point* (such as authentication responsibility) or *part* (such as components without up-to-date security patches) is a reponsibility or component that has a weakness that can be exploited by an attacker. An *exploit path* is a malicious scenario path that involves at least one vulnerable point or part. Figure 4 shows the MUCM notation, which uses black symbols for regular behaviours and red ones for attacks. Exploit path starts with a red triangle instead of a black circle. An unsuccessful exploit, that does not result in damage, ends in a red bar. A successful exploit ends in a red lightning symbol. Like in UCMs, several scenarios and exploit paths can be shown in the same diagram, and exploit paths can be numbered (as will be shown in Figure 5) to represent complex intrusions, in which exploit path builds on the results of previous, successful ones.

Figure 5 shows a MUCM of the bank hack case from Table 1. The MUCM depicts the architecture of the banking computer system, along with an exploit path for each of the main intrusion steps from Table 1. For example, step 7 shows how the attacker exploits the *Admin* use case of Figure 1 to install a *keylogger* on the *Citrix server* to sniff the *Admin password*. In the subsequent step 9, the hacker exploits the *Domain admin* use case to download the *Bank administrator's password hash*.

MUCMs can also show *mitigated vulnerabilities* that help to counter the threats translate to *security requirements*. A hexagon with a curved arrow can show a *searching* the attacker (e.g., searching files for the word “password”). *Get* and *put* arrows show how an attacker interacts with a component, e.g., when the attacker *gets* a copy of a file or *sets* a firewall rule or installs a sniffer program on a server. A stylised *sand-glass* shows that the attacker has to wait for an event, whereas a *question-mark* denotes missing or unclear details.

Ontological analysis of MUCM: Considering the vulnerability-centric ontology for security modelling presented in [81], MUCM covers it quite well (ontology concepts given in bold in next sentence). **Vulnerability** is covered by vulnerable points and parts, **Concrete Element** being the associated component in the architecture and the resource (e.g. information) it contains. **Malicious Action**, consisting of one or more **Attack**(s), can be seen as a number of related exploit paths. The **Effect** can be shown as damage (e.g., lightning symbol). The parts of the ontology that are not so well covered are the **Attacker, Malicious Goal,** and **Affected Element** (related to the

Effect). The presence of an attacker is implicitly understood in the MUCM diagram, but there is no explicit modelling of the attacker as such (type of attacker, what competencies he/she has, etc.). Also, while showing the attack in terms of its exploit path, the diagram may not contain any explanation of the overall goal of the attack (hence, this would likely be left to the accompanying textual explanation). Similarly, the **Affected Element** in [81] may be a function, activity, goal or quality. A MUCM usually does not show which business function or goal is harmed by the attack, only which component it affects and which information may be obtained by the attacker. Hence, the overall purpose, both of the attack and of the attacked system, will not be so clear in a MUCM and would be better covered by other representations, such as goal-oriented modelling techniques. Indeed, a misuse case diagram (MUC) also gives a better view of the purpose than a MUCM does, showing the use cases (what the system is supposed to achieve) and the misuse cases (what the attacker tries to achieve), as well as explicitly showing the attacker in the diagram, with a name label indicating the type of attacker. On the other hand, MUC does not have the same coverage of the **Vulnerability** and the **Concrete Element** (e.g., architecture component) that it relates to. All in all, thus, it can be seen that MUCM has some strengths in showing detailed relationships between vulnerabilities, attack sequences, and architecture components, but also weaknesses in terms of poorer coverage of goals and functions where, e.g., goal-oriented or process-oriented techniques will be stronger. MUCM is therefore not suggested to replace all other techniques, but rather to be used together with other techniques.

Formative evaluation of MUCM: [16] reports a preliminary evaluation of MUCM. A written evaluation form was sent out the MUCM notation to more than 20 colleagues and other contacts and 12 responses were received. All respondents had M.Sc or Ph.D degrees in computing, except one M.Sc student who, on the other hand, had professional experience as a system administrator. 6 of the respondents were working in academia, 4 in industry, and 2 in both academia and industry. The evaluation sheet first explained the aim of and the required conditions for the experiment. It then gave an introduction to UCM and MUCM, before presenting the textual description of the bank hack described along with the MUCM shown in Figure 5. Three sets of questions were then posed regarding (a) the backgrounds of the participants, (b) the participants' understanding of the case, and (c) the user's acceptance of the technique. We also asked the participants which aids they had relied on when answering questions about understanding of the case and how much time they had used. Finally, open comments were invited.

The responses were analysed according to which aids each participant reported to have relied on for the questions about understanding. The *textual group* contained 9 valid responses relying on the textual description. The *memory group* had 6 valid responses relying on memory, and the *MUCM group* comprised 6 valid responses relying on the misuse case map. Finally, the *non-MUCM group* contained 4 valid responses from participants who did not use the misuse case map. With the exception of the mutually exclusive MUCM and non-MUCM groups, the four overlapped considerably because most respondents had relied on more than one aid. Nevertheless, the comparison of responses gave a useful first indication of the strengths and weaknesses of MUCM. In particular, the MUCM group achieved the highest mean score on understanding, whereas the non-MUCM group scored lowest. Along with the textual group, the MUCM group also spent more time on the questions than the memory and non-MUCM groups. All groups rated the usefulness of MUCM similarly, but the MUCM group rated perceived ease of use higher than the non-MUCM group, which in turn scored highest on intention to use MUCMs in the future. The background expertise of the groups was similar, although the TD and MUCM groups reported slightly less experience.

Comparison of MUCM with misuse sequence diagrams: [21] presents an experimental comparison between MUCM and another proposed security technique: *misuse sequence diagrams (MUSD)*, which represents the sequence of attacker interactions with system components and how they were misused over time by exploiting their vulnerabilities. The authors investigated MUSD and MUCM in a controlled experiment with 42 students. Their comparison indicated that they are complementary in terms of understanding. They aid understanding of different aspects about intrusions into systems, i.e., the architecture (MUCM) and the sequence of events, actions (MUSD). They are equal in terms of performance, i.e., they encourage users to identify similar numbers of vulnerabilities and mitigations in the planned system. However, MUSD was perceived more positively by users, i.e., they rate the technique more highly in terms of perceived usefulness, perceived ease of use, and intention to use. The difference is most marked for perceived ease of use.

The results from these preliminary evaluations have encouraged us to prepare more extensive and controlled evaluations of MUCM, the results of which we present in this paper. On the positive side, the integrated model makes a clearer connection between security and architecture, but it does so at the cost of making the visual notation more complex and potentially harder for stakeholders to understand, compared to showing security and architecture in two separate diagrams. Indeed, these are then two potentially counteracting principles when considering the quality of visual notations, called *cognitive integration* and *complexity management* in [82].

3. Research method

3.1 **Treatments**

The purpose of our study was explained already in the introduction: we want to evaluate misuse case maps experimentally. The *treatment group* in our experiments used *misuse case maps (MUCM)* to carry out a sequence of experiment tasks (to be described below). The *control group* instead used an alternative technique (the *null treatment*) to carry out the same tasks. As we explained in the previous section, there are no other techniques that focus specifically on representing security threats, vulnerabilities, and mitigations in a software systems architecture context. We therefore chose combined use of two techniques as alternative (or null) treatment: *misuse case diagrams (MUC)* and *system architecture diagrams (SAD)*, which were both explained in Section 2.5.

We chose this particular combination of techniques because: (1) MUC and SAD are both well-established and practical techniques and therefore a natural choice for developers who want to represent both security and architectural concerns, and (2) using MUC and SAD together is the alternative we have found that best approximates MUCM, without introducing new or overly complex notations just for the purposes of our experiment. We therefore took care to make the SAD display architecture just like the MUCM, but with responsibilities, vulnerabilities, and exploitation paths removed. Accordingly, we took care to make the MUC display the same misuse behaviour as the MUCM, but with the architectural context removed.

We will refer to the experimental treatment as *MUCM* and the alternative treatment as *MUC-SAD* in the rest of the paper. We will return to discuss the differences between them when we discuss internal validity in Section 5.4.

3.2 Research questions

The specific purpose of our experiments was to evaluate whether misuse case maps facilitate *good understanding* of intrusion scenarios, whether they support *effective identification* of security issues – like threats, vulnerabilities, and mitigations – and whether they are *perceived positively* by their users. On the positive side, MUCM provides a clearer connection between security and architecture than can be done by separate architecture and threat models. On the negative side, the visual notation becomes more complex and potentially harder for stakeholders to understand, compared to dedicated notations for architectures and threats. Indeed, these are potentially counteracting principles when considering the quality of visual notations, called *cognitive integration* and *complexity management* in [82].

To investigate them further, we therefore formulated three research questions:

- RQ1: Do misuse case maps aid understanding of intrusions better than the alternative treatment?
- RQ2: Do misuse case maps aid identification of more security issues than the alternative treatment?
- RQ3: Are misuse case maps perceived more positively by users than the alternative treatment?

To investigate possible relationships between understanding, identification, and acceptance, we also asked:

- RQ4: Does a technique[2] that better aids understanding of intrusions also aid identification of more vulnerabilities and mitigations?
- RQ5: Is a technique that better aids understanding of intrusions also perceived more positively by its users?
- RQ6: Is a technique that aids identification of more security issues also perceived more positively by its users?

[2] Here and later, to keep our language plain, we will talk about the alternative treatment as "a technique", although it is strictly "a pair of techniques".

3.3 Experiment designs

We have run two experiments to answer these research questions (we will explain below why we chose to run two). In each experiment, we compared the understanding, ability to identify security issues, and perceived technique acceptability of participants who used misuse case maps with those of participants who used another technique.

Experiment 1: The first experiment was a simple comparison experiment that involved 33 participants recruited from a class of 3rd year ICT students. The students were recruited through their student organization, which received some payment for each participant volunteering for the experiment. One half of the participants, the treatment group (17 participants), performed a given sequence of tasks using MUCM. The other half, the control group (16 participants), used MUC-SAD. Both groups carried out the tasks once using their assigned technique. As intrusion case, we used the bank hack case [80], chapter 7 that we described in Section 2.5. The participants went through the following steps:

1. Answering a questionnaire asking for background information about the participant (3 min).
2. Reading a short introduction to the experiment (1 min).
3. Reading a short introduction to the assigned technique (1.5 pages, 11 min).
4. Reading an intrusion case described both textually and as a diagram on a separate page (5 pages, 18 min).
5. Judging the truth or falsity of 20 statements about the intrusion, with both the text and the diagram available (9 min).
6. Suggesting vulnerabilities and mitigations related to the intrusion, with both the text and the diagram available (17 min).
7. Answering a questionnaire about how the participant perceived the technique (4 min).

The total duration of the first experiment was therefore a little over an hour. Although it produced many useful results, Sections 4. and 5. will reveal that the first experiment had two weaknesses:

- Power analysis showed that the effects of using MUCM over MUC-SAD were a little weak for an experiment with only 33 participants. In the second experiment, we therefore attempted to recruit more participants, and we used a full factorial (Latin-squares or crossover) design to allow stronger statistical testing.
- The first experiment did not control for whether the participants solved the experiment tasks using the textual description or the diagram descriptions we provided. In other words: we wanted to compare MUCM with MUC-SAD, but we ended up comparing MUCM and a textual description with MUC-SAD

and the same textual description, which is a less sharp comparison. In the second experiment, we therefore controlled for diagram use.

Experiment 2: The second experiment was therefore a *crossover experiment* that involved 54 participants recruited from another class of 2nd year ICT students, which were again recruited through their student organization. In the crossover experiment, each participant carried out the above activities *twice* (except for the background questionnaire, which was answered only once by each participant): once using the experimental treatment (MUCM) and once using the alternative treatment (MUC-SAD). Half the participants used MUCM first and then MUC-SAD, the other half used MUC-SAD first and then MUCM. To reduce the effect of learning about the case, different intrusion cases were used in the two parts of the experiment: the bank hack case (BANK) from the first experiment along with a penetration test case (PEN), described in [80], chapter 6. Half the participants worked on the BANK case first and then PEN, the other half worked on PEN first and then BANK. The participants were thus assigned randomly into the four equally-sized groups (A-D) shown in Table 2, corresponding to the four possible combinations of treatment sequences MUCM/MUC-SAD and BANK/PEN. The participants went through the same 7 steps as in the first experiment with the following differences:

- Steps 3-7 were carried out twice, once using the first assigned technique on the first assigned case, then using the second technique on the second case. The participants had a short break between these two experiment periods. This extension was necessary to implement the crossover design.
- In steps 5 and 6, the participants were only allowed to use the provided diagrams, not the textual descriptions of the provided case. This modification was necessary to control for diagram use.
- After steps 5 and 6, the participants were asked to report how much they had relied on the provided diagrams relative to memory and other aids. This extension was necessary to control for diagram use.

The second experiment thus lasted almost 2 hours. In addition to RQ1-RQ6 from the first experiment, we added the following research questions about diagram use:

- RQ7: Are misuse case maps used more than the alternative treatment when solving understanding and identification tasks?
- RQ8: Does a technique that encourages more diagram use also aid better understanding of intrusions?
- RQ9: Does a technique that encourages more diagram use also aid identification of more security issues?
- RQ10: Is a technique that encourages more diagram use also perceived more positively by users?

Arrangements: In both experiments, the participants worked individually under equal conditions in the same room and at the same time. We placed the participants in the room so that participants in the same group were never seated next to one another. The duration of the different steps was decided dynamically during the experiment, so that the participants were given enough time to finish each step. The exception was step 5, which we always stopped after 9 minutes – before everyone could finish – to collect data about how efficiently each participant was able to use the techniques.

3.4 Variables

We used the following *independent variables* to describe each experiment and each part (or period) of the second experiment (Table 3):

- **Experiment (EXP):** Either the first (EXP1) or second (EXP2) experiment.
- **Technique (TECH):** Either the experimental treatment (MUCM) or the alternative treatment (MUC-SAD).
- **Case (CASE):** Always the bank hack case (BANK) in the first experiment. Either BANK or the penetration test case (PEN) in the second experiment.
- **Period (PERIOD):** Always PERIOD1 in the first experiment, which had only one period. Either PERIOD1 or PERIOD2 in the second experiment, where a period corresponded to using one of the techniques (MUCM or MUC-SAD) on one of the intrusion cases (BANK or PEN).

At the start of each experiment, each participant filled in a *background questionnaire*, which collected data about the following *control variables* for each participant:

- **Knowledge (KNOW):** In the first experiment, each participant was asked to assess his or her knowledge in the following areas: system modelling (KNOW_MOD), use case maps (KNOW_UCM), misuse cases (KNOW_MUC), and misuse case maps (KNOW_MUCM). In the second experiment, they were also asked to assess their knowledge in system architecture diagrams (KNOW_SAD) and security threat analysis (KNOW_SAD). In the second experiment, we added questions about threat modelling (KNOW_THR) and use cases (KNOW_UC). Each knowledge item was measured on a 5-point Likert-like scale, where 1 was "Never heard of it" and 5 was "Expert".
- **Semesters of ICT-study (ICT_STUDY):** In both experiments, each participant was asked to report how many semesters he or she had studied ICT.
- **ICT-related work experience (ICT_WORK):** In both experiments, each participant was asked to report how many years of ICT-related work experience he or she had.

Our research questions involved the following three groups of *dependent variables:* understanding of intrusions (UND), identification of vulnerabilities and mitigations (IDEN), and perception (or acceptability) of the technique used (ACC). We designed our experiments to measure these variables in the following ways for each period of each experiment:

- **Understanding of intrusions (UND):** We let the participants read an intrusion case from the literature, described both textually and diagrammatically. We then presented them with an *understanding questionnaire*, which asked them to judge 20 statements about the case as either true or false. Ignoring missing judgments, we counted the number of correct minus the number of wrong answers (UND). We also counted the number of judgments each participant made in the available time (UND_ANS). A final measure was the number of correct minus the number of wrong answers, counted only for those participants that had managed to judge all 20 statements within the allocated time (UND_ALL).
- **Identification of vulnerabilities and mitigations (IDEN):** We asked the participants first to identify as many vulnerabilities as possible to the intrusion – a central security requirement task [28] – and then to identify as many mitigations as possible to those vulnerabilities, possibly several mitigations per vulnerability. We did not ask them to focus on specific types of vulnerabilities or mitigations. Afterwards, we counted the numbers of unique vulnerabilities (IDEN_VUL) and mitigations (IDEN_MIT) found, as well as their sum (IDEN). Although type and criticality of the vulnerabilities and mitigations are also important, they were out of scope of the experiments reported in this paper and have to be left for future work.
- **Perception of the technique used (ACC):** We let the participants fill in an *acceptance questionnaire* about how they perceived their assigned technique. The questionnaire used 12 statements, or items, from the *technology acceptance model* TAM [83]. 4 of the items were intended to measure perceived usefulness (ACC_PU) of the technique used, 4 were intended to capture ease of use (ACC_PEOU), and 4 to measure intention to use (ACC_ITU) the technique again in the future. The items for ACC_PU, ACC_PEOU, and ACC_ITU were presented in mixed order. Each one was scored on a 5-point Likert-like scale, where 1 was "Strongly disagree" and 5 was "Strongly agree". One statement in each group was phrased negatively, so that a high score would indicate little perceived usefulness. We chose the original version over its later modifications and extensions [84], [85] because it offers thoroughly validated, well-known and widely used instruments for user acceptance that suited our need and because our study does not aim for the more precise prediction of future use offered by TAM2 [84] and TAM3/UTAUT [85].

Whereas UND and IDEN were thus measured by objective *performance variables*, ACC instead involved subjective *perception variables*. The second experiment introduced a fourth *dependent variable*:

- **Diagram use (DIAG):** In the second experiment, but not the first, we asked the participants to estimate – using a percentage – how much they had relied on the provided diagrams when solving the understanding task (DIAG_UND) and when solving the identification tasks for vulnerabilities (DIAG_VUL) and mitigations (DIAG_MIT). For DIAG_UND and DIAG_VUL, we asked them to estimate the relative use of diagrams compared to use of memory, so that, e.g., a score of 0% for DIAG_UND would indicate that the 20 statements had been judged only using memory, whereas a score of 100% for DIAG_VUL would indicate that vulnerabilities had been identified using only information from the provided diagrams. For DIAG_MIT, we asked them to estimate diagram use compared to use of either memory or of the already identified vulnerabilities.

3.5 Coding

We coded the data we collected as follows.

Background questionnaire: The background questionnaire had only closed or numerical items. The self-assessed knowledge levels were already numerical on a scale from 1 (“Never heard of it”) to 5 (“Expert”). The self-reported prior ICT-studies in semesters and ICT-related work experience in years were also already numerical.

Understanding questionnaire: Each correct judgment was scored as 1 and each incorrect judgement counted -1. Missing judgements were scored as 0. For each participant, the scores for all 20 statements were added up (UND) and the number of non-zero scores counted (UND_ANS). We also added up the scores for all 20 statements only from those participants who had judged all 20 statements (UND_ALL).

Vulnerability and mitigation sheets: We counted the numbers of unique vulnerabilities and mitigations that each participant had identified. Because each mitigation was associated with a particular vulnerability, we also counted the numbers of vulnerabilities for each mitigation. Because the focus in early requirements engineering is on identifying possibilities, we did not attempt to assess the severity, likelihood, relevance, feasibility or other aspects of the vulnerabilities and mitigations identified.

Diagram use: The self-assessed estimates of diagram use were already in numerical form as percentages.

Acceptance questionnaire: The answers to the 12 questions about how the given technique was perceived were already numerical on a scale from 1 to 5. All answers to the three negatively phrased questions were reversed on this scale ($x_{new} = 6 - x_{old}$). For each participant, we calculated mean scores for each group of four questions about perceived usefulness (ACC_PU), about perceived ease of use (ACC_PEOU), and about intention to use again (ACC_ITU). For each participant, we also calculated overall mean scores for all the 12 questions (ACC).

3.6 Hypotheses

To allow for statistical testing of our data, we used the above variables to formulate a research hypothesis for each research question. The *alternative form* of each hypothesis describes how we expect the treatment (MUCM versus MUC-SAD) to produce a different outcome. Table 4 also shows the corresponding *null hypotheses* (that our treatments had no effect) that were actually used in the statistical tests. Hypotheses H1-H3 correspond to RQ1-RQ3:

- H1: Participants using MUCM will understand an intrusion better than participants using MUC and SAD separately.
- H2: Participants using MUCM will identify more security issues for an intrusion than participants using MUC and SAD separately.
- H3: Participants using MUCM will perceive their technique differently from participants using MUC and SAD separately.

Hypotheses H1 and H2 were directional in the favour of MUCM based on our preliminary results and expectations, whereas hypothesis H3 was un-directional. We also formulated the following hypotheses H4-H6 to answer research questions RQ4-RQ6:

- H4: Participants who understand an intrusion better using a technique will also identify more security issues for that intrusion using the technique.
- H5: Participants who understand an intrusion better using a technique will also perceive that technique more positively.
- H6: Participants who identify more security issues for an intrusion using a technique will also perceive that technique more positively.

In the second experiment, we also formulated the additional H7-H10 hypotheses to answer RQ7-RQ10:

- H7: Participants using MUCM will use diagrams more than participants using MUC and SAD.
- H8: Participants who use diagrams more when using a technique will also better understand the intrusion using that technique.
- H9: Participants who use diagrams more when using a technique will also identify more security issues for the intrusion using that technique.
- H10: Participants who use diagrams more when using a technique will also perceive that technique more positively.

4. Hypothesis testing

Shapiro-Wilk tests showed that none of our main variables were normally distributed (alpha=0.05) in the second experiment. In the first experiment, a few main variables (UND, IDEN_x, ACC_ITU, ACC) were normally distributed, but a few of them barely above alpha (0.05) and several others below. To be able to use the same tests for all variables in both experiments, we therefore used non-parametric statistical tests throughout. We used exact significances and a confidence level of 0.05. We used one-tailed statistical tests for the directional hypotheses (H1-H2, H7) and two-tailed tests for the un-directional one (H3) and for the correlation hypotheses (H4-H6, H8-H10).

4.1 Participant backgrounds

Tables 5 and 6 describe the backgrounds of the participants in the first and second experiments, respectively. For each control/background and dependent variable, Table 5 first shows the number of observations for the variable (*n*). In the first experiment, this is 33 for most variables, although one participant did not perform the identification tasks and only 23 answered all the understanding questions. The table then shows the *median* as well as the *mean*, standard deviation (*st.dev*), and minimum (*min*) and maximum (*max*) observations for the dependent variable. For each control/background and dependent variable, Table 6 shows the same information as Table 5. The numbers of observations are again equal to the numbers of participants for the control/background variables, but for the dependent variables, there are up to twice as many observations as participants, because each participant carried out the experimental tasks twice.

In the first experiment, the participants reported between 2 and 5 semesters of ICT studies (median 5) and between 0 and 36 months of ICT-related work experience (median 1). In the second experiment, the participants reported between 2 and 7 semesters of ICT-studies (median 3) and between 0 and 10 months of ICT-related work experience (median 0, meaning that a majority of the participants had no work experience).

We compared the backgrounds of the participants in the first and second experiments using Wilcoxon rank-sum testing, a non-parametric test of whether two independent samples originate from the same distribution (thus it is a non-parametric alternative to the Student t-test for two independent samples). Not surprisingly, the difference in semesters of ICT study between the two groups was significant, but not the difference in ICT-related work experience. The mostly fifth-semester students in the first experiment reported being significantly more knowledgeable in all knowledge areas than the mostly third-semester students in the second experiment. However, the difference between the two experiment groups has not impacted our main results, which are primarily derived from analysing the two experiments separately, and sometimes pooling the data, but never comparing the two sets of participants directly.

Table 7 compares the backgrounds of the two experiment groups (MUCM and MUC-SAD) in the first experiment using Wilcoxon rank-sum (also called Mann-Whitney U) tests. For each control/background and dependent variable, Table 7 shows three types of information indicated in the topmost row: a summary of the *MUCM* and of the *MUC-SAD* observations, results of the Wilcoxon rank-sum test (*W. rank-sum*), and the *effect* of the MUCM versus MUD-SAD treatments. For each MUCM and MUC-SAD observation, the numbers of observations (*n*) are shown along with the *mean* and standard deviation (*St.dev*). These data are thus comparable to those of Table 5, but with MUCM and MUC-SAD observations separated. The Wilcoxon rank-sum tests are summarised with the V metric (returned from the *wilcox.test* package of the statistical analysis tool *R*) and the significance level (*p*). The effect on each dependent variable of using MUCM instead of MUC-SAD is indicated using *Cohen's d* along with the sample size that would have been required to make the results significant at the 0.05 level (*Req.sample*) and with the effect required to make the results significant (again on the 0.05 level) with the given sample size (*Req.eff.*). There were no significant differences between the backgrounds of the two groups, except for previous ICT-work experience, which we will discuss along with threats to validity in Section 5.4.

Table 8 compares the backgrounds of the four groups in the second experiment using Kruskal-Wallis testing, an extension of non-parametric Wilcoxon rank-sum testing to more than two independent samples (and thus a non-parametric alternative to the Student t-test for more than two independent samples). For each control/background variable, Table 8 again shows five types of information indicated in the topmost row summaries of experiment groups *A*, *B*, *C*, and *D* and results of the *Kruskal-Wallis* test. For each group, the numbers of observations (*n*) are shown along with the *mean* and standard deviation (*St.dev*). These data are comparable to those of (the top half of) Table 6, but with the observations separated by groups. The Kruskal-Wallis tests are summarised with the standard *K* metric and the significance level (*p*). Again, the only significant difference was for previous ICT-work experience, which we will discuss along with threats to validity in Section 5.4.

In the first experiment, the participants reported being significantly more knowledgeable about systems modeling than about analysis of security threats (means: KNOW_MOD=2.88, KNOW_SEC=2.21, p=0.002) and more knowledgeable about misuse cases than about misuse case maps (avg. KNOW-MUC=2.21, KNOW_MUCM=1.76, p=0.003). In the second experiment, we also asked the participants to assess their knowledge about use cases and system architecture modelling. They again reported being significantly more knowledgeable about systems modeling than about analysis of security threats (means: KNOW_MOD=2.63, KNOW_SEC=1.78, p=0.000) and a little more knowledgeable about misuse cases than about misuse case maps, but not significantly so (avg. KNOW_MUC=1.22, KNOW_MUCM=1.13, p=0.073). They also reported being significantly more knowledgeable about use cases than about use case maps (KNOW_UC=3.02, KNOW_UCM=2.11, p=0.000) and significantly more knowledgeable about system architecture modelling than about misuse case maps (means: KNOW_SAD=2.49, KNOW_MUCM=1.13, p=0.000).

4.2 Understanding

Experiment 1: We performed Wilcoxon rank-sum tests to compare the results of the participants in the independent MUCM and MUC-SAD groups on the understanding task in the first experiment. Table 7 shows that the MUCM group had significantly better understanding than the MUC-SAD group both when considering all participants (UND[MUCM]=15.35, UND[MUC-SAD]=12.5, p=0.011) and when ignoring those participants who did not judge all the 20 statements (UND_ALL[MUCM]=16.17, UND_ALL[MUC-SAD]=12.55, p=0.010). The MUCM group also judged more of the 20 statements (whether correctly or incorrectly) than the MUC-SAD group (UND_ANS[MUCM]=19.24, UND_ANS[MUC-SAD]=18.38, p=0.33), but the difference was not significant.

The effect sizes (Cohen's d [86]) were 0.82 and 1.14 respectively, for all the participants (UND) and for only those who assessed all 20 statements (UND_ALL). Cohen [86] classifies these effects as large (>0.8) and Hopkins [87] as moderate (> 0.6).

Experiment 2: We compared the results of the understanding task when the participants were using MUCM and when they were using MUC-SAD in the second experiment using Wilcoxon signed-ranks testing, a non-parametric test of whether two paired samples originate from the same distribution (thus a non-parametric alternative to the Student t-test for two paired samples). For each dependent variable, Table 9 shows the same information as did Table 7. For each MUCM and MUC-SAD observation, the data are again comparable to those of (the bottom half of) Table 6 with the MUCM and MUC-SAD observations separated.

Table 9 shows that MUCM facilitated significantly better understanding than the alternative treatment, both when considering all participants (UND[MUCM]=12.91, UND[MUC-SAD]=9.87, p=0.000) when only considering those participants who judged all the 20 statements (UND_ALL[MUCM]=12.91, UND_ALL[MUC-SAD]=9.87, p=0.000). There was no significant difference in how many participants judged all of the 20 statements (whether correctly or incorrectly). The effect sizes were 0.71 and 0.83 which are, respectively, medium (> 0.5) and large according to Cohen's classification [86] and moderate according to Hopkins's [87].

Pooled data: We also performed Wilcoxon signed-ranks tests of the participants' understanding when using MUCM and when using MUC-SAD using *pooled* (or *merged*) data from both experiments. Before pooling, we shifted (linearly transformed) the data from the second experiment so that the mean of each variable from the second experiment became identical to the mean of that variable from the first experiment, but so that the transformation did not alter variance. The reason we transformed the data like this, was to cancel out the effect of having significantly more knowledgeable and experienced participants in the first experiment than in the second. The analysis confirmed that understanding was significantly better when using MUCM (p=0.000). But there were no significant differences for numbers of statements judged or when considering only the understanding of those participants who judged all 20 statements.

Conclusion: Both experiments suggest that MUCMs aids understanding significantly better than MUC-SAD. The difference remains significant when we only use the results from those participants who worked efficiently enough to assess the truth or falsity of all the 20 statements. Similar numbers of participants were able to assess all 20 statements whether they were using MUCM or MUC-SAD, suggesting that the difference in understanding is due to quality of comprehension rather than speed of comprehension. We therefore accept hypothesis H1.

4.3 Identification

Experiment 1: We performed Wilcoxon rank-sum tests to compare the results of the MUCM and MUC-SAD groups on the identification task in the first experiment. Table 7 shows that the MUCM group found significantly more mitigations than the MUC-SAD group (IDEN_MIT[MUCM]=8.56, IDEN_MIT[MUC-SAD]=4.87, p=0.015). The MUCM group also found somewhat more vulnerabilities than the MUC-SAD group (IDEN_VUL[MUCM]=7.94, IDEN_VUL[MUC-SAD]=7.12, p=0.28), but the difference was not significant. The difference in total number of vulnerabilities and mititations found (IDEN) remained significant (p=0.019). The respective effect sizes (Cohen's d) were 0.30, 0.88, and 0.77. These are small (> 0.2), large and medium according Cohen [86], and small (> 0.2), moderate and moderate according to Hopkins [87].

Experiment 2: We performed Wilcoxon signed-ranks tests to compare the results of the identification task when the participants were using MUCM and when they were using MUC-SAD in the second experiment (Table 9). However, there were no significant differences.

Pooled data: We again performed Wilcoxon signed-ranks tests using the pooled data from both experiments. There were no significant differences between the participants' performance on the identification tasks when using MUCM and when using MUC-SAD.

Conclusion: Significantly more mitigations were identified using MUCM in the first experiment, but in the second experiment there were no significant differences. We therefore reject hypothesis H2. Further research is needed to explain why the two experiments gave different results.

4.4 Perception

Experiment 1: We performed Wilcoxon rank-sum tests to compare how the MUCM and MUC-SAD groups perceived the techniques they were assigned in the first experiment. Table 7 shows that there were no significant differences, neither overall (ACC), nor for the specific measures perceived usefulness (ACC_PU), perceived ease of use (ACC_PEOU) or intention to use (ACC_ITU) the technique, although the group using MUC-SAD perceived their technique a little more positively than the MUCM group for each measure.

Experiment 2: We performed Wilcoxon signed-ranks tests to compare how the participants reported perceiving the MUCM and MUC-SAD techniques in the second experiment. Table 9 shows that MUCM diagrams were significantly more positively than the combined use of MUD and SAD diagrams, both for perceived usefulness

(ACC_PU, p=0.000), perceived ease of use (ACC_PEOU, p=0.008), intention to use (ACC_ITU, p=0.000), and overall (ACC, p=0.000). The respective effect sizes were 1.24, 0.44, 0.82, and 0.94. According to Cohen [86], this means they are all large, except ACC_PEOU, which is small. According to Hopkins [87], they are large (> 1.2), small, moderate and moderate, respectively.

Pooled data: We again performed Wilcoxon signed-ranks tests using the pooled data from both experiments. There were significant differences both between ACC_PU, ACC_ITU, and ACC_PEOU (p=0.000, p=0.005, p=0.000, respectively) between the participants' performance on the identification tasks when using MUCM and when using MUC-SAD.

Conclusion: The participants perceived MUCM significantly more positively in the second experiment, but in the first experiment there were no significant differences. However, when pooling the data from both experiments, the difference was significant both overall and for ACC_PU and ACC_ITU. We therefore accept hypothesis H3. Further research is needed to explain why they two experiments gave different results.

4.5 Correlations between understanding, identification, and perception

For both experiments, we also performed correlation analyses using Spearmann's rho between our main variables: understanding (UND), number of vulnerabilities (IDEN_VUL), and mitigations (IDEN_MIT) identified, both separately and overall (IDEN) as well as perceived usefulness (ACC_PU), perceived ease of use (ACC_PEOU), intention to use again (ACC_ITU), and overall perception (ACC). In the rest of the paper, correlation will mean correlation with significance $p < 0.05$ and strong correlation will mean significance $p < 0.01$.

Experiment 1: In the first experiment, we performed correlation analyses both of the MUCM and MUC-SAD groups separately and of the pooled data from both groups. For the pooled data (MUCM and MUC-SAD together), Table 10 there were few correlations between the dependent main variables. The only ones we found were that understanding (UND) was correlated positively both with identifications of mitigations (IDEN_MIT) and overall (IDEN) ($0.50 < rho < 0.55$). Surprisingly, the main variables were not always correlated even within their respective groups.

Considering the data from the MUCM and MUC-SAD groups separately confirmed this picture. There were even fewer correlations in the data from the MUC-SAD group. The correlations tended to be weaker, and understanding in this group was no longer correlated with any other main variable. The only exception was that vulnerabilities identified was correlated with intention to use in the MUC-SAD group but, with so many variables analysed, there are bound to be false positives, so we should be careful when we draw conclusion about correlations between other variables than UND, IDEN, and ACC.

The correlations also suggested an explanation why the relative performance of the MUCM group was better for mitigations than for vulnerabilities. Although the numbers of vulnerabilities and of mitigations were not significantly correlated for the pooled sample, when analysing the two groups separately, the correlations became significant for the MUCM group (rho = 0.53, p = 0.017), but not for the MUC-SAD group (rho = -0.019).

Experiment 2: In the second experiment, we performed correlation analyses both of the pooled data for the two experiment periods (PERIOD1 and PERIOD2) taken together and of the MUCM and MUC-SAD data (from both periods) separately.

For the pooled data (both MUCM and MUC-SAD together), Table 11 shows that almost all the main variables were correlated with one another ($p < 0.05$), many of them strongly ($p < 0.01$). Not surprisingly, understanding was strongly and positively correlated ($0.25 < rho < 0.35$) with all the other main variables, except intention to use, which was only correlated with other perception variables. Perceived usefulness was also not correlated with the performance variables, whereas perceived ease of use was strongly and positively correlated ($0.30 < rho < 0.40$) with the understanding and performance variables. This suggests that ACC_PEOU is the better indicator of both understanding and performance in experiments, which is perhaps not surprising, given that ease of use of a technique is central in a limited-time/limited-problem situation such as an experiment, whereas perceived usefulness and intention to use has to do with matters outside of the experiment.

Considering only the data when MUCM was used, almost all the main variables were again correlated. With this subset of data, however, ACC_ITU was correlated positively to performance ($0.25 < rho < 0.40$), but there was no significant correlation between ACC_PU and performance. Considering only the data when MUC-SAD was used, there were few correlations between the main variable types: understanding was correlated with identification of vulnerabilities (rho=0.32) and perceived ease of use was correlated with performance ($0.25 < rho < 0.35$).

Pooled data: We also performed a correlation analysis using the pooled data from both experiments. All the main variables very strongly and significantly correlated.

Conclusion: In the second experiment, the correlations are significant and strongly positive as expected for the pooled data. When we analyse the MUCM and MUC-SAD data separately, we find the same pattern of correlations reflected in the MUCM data, but less clearly so in the MUC-SAD data. In the smaller first experiment, we do not find many correlations between the variable groups, but there are some, and the tendency remains that the correlations we find in the pooled data are reflected more strongly in the MUCM data than in the MUC-SAD data. We therefore accept hypotheses H4-H6.

4.6 Use of diagrams

Experiment 2: We performed Wilcoxon signed-ranks tests to compare the how much the participants reported using the provided diagrams for solving their tasks when using MUCM and when using MUC-SAD in the second experiment. Table 9 shows that the participants used the MUCM diagrams significantly more than the MUD and SAD diagrams, both when solving the understanding task (p=0.000) and when identifying vulnerabilities (p=0.000) and mitigations (p=0.003). The effect sizes were 1.07, 1.32, and 0.35, respectively, which means large, large and small in Cohen's terms [86], and moderate, large and small according to Hopkins [87]. The weaker effect for mitigations can perhaps be explained by the nature of the task: when finding mitigations, the participants were investigating further the vulnerabilities they had already identified themselves, making them somewhat less dependent on documentation in the form of diagrams. Both when solving the understanding tasks and when identifying vulnerabilities and mitigations, the participants reported relying significantly more on memory when using MUC-SAD.

Conclusion: The participants used the MUCM diagrams significantly more often for all three tasks. We therefore accept hypothesis H7.

4.7 Correlations involving diagram use

Experiment 2: In the second experiment, we also performed correlation analyses using Spearmann's rho between our main variables, both of the pooled data for the two experiment periods (PERIOD1 and PERIOD2) taken together and of the MUCM and MUC-SAD data (from both periods) separately.

For the pooled data (both MUCM and MUC-SAD together), Table 11 shows that all the three diagram use variables were correlated with understanding ($0.15 < rho < 0.35$). They were also strongly correlated with all the perception variables ($0.30 < rho < 0.60$), except with perceived ease of use. The strongest correlations of diagram use were instead with perceived usefulness (p=0.000, $0.35 < rho < 0.60$).

Considering only the data when MUCM was used, diagram use is no longer correlated with understanding and the correlation of ACC_PU and DIAGMITI is lost, but both DIAGUND and DIAGVULN are correlated with the performance variables ($0.25 < rho < 0.35$). Considering only the data when MUC-SAD was used, even more correlations are lost and no new ones appear.

All three analyses (pooled and MUCM and MUC-SAD separately) showed that diagram use for understanding, for identifying mitigations, and for identifying vulnerabilities were always correlated. The weakest correlations were always between diagram use for understanding and for identifying mitigations, perhaps because the already identified vulnerabilities were also used much in the latter case. All the other correlations were strong ($p < 0.005$,

$0.40 < rho < 0.65$). Hence, those participants who used the diagrams in one of the tasks would tended to use it for the other tasks, too, suggesting that personality typing at the pre-experiment questionnaire might be interesting in order to find out who these consistent diagram users are.

Conclusion: Our analyses show clearly that diagram use is significantly correlated with both understanding and with identifying vulnerabilities and mitigations. We therefore accept hypotheses H8 and H9.

4.8 Controlling for carryover effects

The Latin-Square or 2x2 crossover design we used in the second experiment was advantageous because it allowed stronger statistical testing of *paired samples* (such as Wilcoxon signed-ranks tests), which are often able to produce significant results with fewer participants than the corresponding tests for *independent samples* (such as Wilcoxon rank-sum tests). However, crossover designs also bring with them a danger that later experiment periods are affected by earlier ones, for example through *learning*, *priming*, *fatigue* or other effects. *Learning* from the first period could have affected the participants' behaviour in the second period in several ways: they could have become more knowledgeable about security intrusions; better at comprehending security intrusions; better at generating vulnerabilities and mitigations; more in-the-know about the purpose of the experiment; etc. *Priming* could have raised or lowered the participants' expectations, e.g., of how useful or easy-to-use a security technique should be. *Fatigue* could have resulted in participants working harder and better in the first period than in the second.

Such effects are often called *carryover effects*, referring to biomedical trials where the treatment administered to patients in an early period could persist to distort later periods. *Washout periods* are therefore typically recommended between treatment periods. In our second experiment, extended washout periods might have helped reduce fatigue (and perhaps to some extent priming), but not learning effects, which may in some cases even grow with time as cycles of action and reflection leads to experiential learning [88]. Hence, a washout period – although intended to have the subjects forget their treatment – might instead provide at least some of them with time for post-task reflection that would increase their ability to solve the experiment tasks. Such a reflection effect would be hugely individual, increasing subjective variation: some students might indeed forget much about the initial treatment during a long washout period, whereas others might increase their understanding of it through post-task reflection and subsequent association of these reflections to other learning experiences. A longer, uncontrolled washout period could also potentially have introduced additional carryover effects, such as the participants sharing information about the techniques and cases they had been using in the first period. Also, some participants might have become especially interested in security-related topics after the first experiment round, thus deciding to learn more by seeking out extra-curricular material.

We therefore chose to design the experiment with only a short, controlled break between the two periods to reduce fatigue, while using the following other means to limit and control for carryover effects in general:

- Firstly, half the participants used MUCM first and the other half MUC-SAD first, so that carryover effects would be counterbalanced between the techniques.
- Secondly, each participant used the two techniques on two different intrusion cases (BANK and PEN) to eliminate effects from learning about the task.
- Thirdly, half the participants considered the BANK case first and the other half the PEN case first, so that carryover effects would also be counterbalanced between the intrusion cases.
- Fourthly, we used several statistical tests (see below) to control for significant differences between the results from the first half of the experiment and the second and between the results from the two intrusion cases.

To control for carryover effects in general, we used Wilcoxon signed-ranks tests for differences between the first and second periods of the second experiment (regardless of technique used). We found no significant differences for understanding, performance or diagram use, suggesting that fatigue or carryover have not strongly affected our results. However, for perceived usefulness (ACC_PU) and intention to use again (ACC_ITU), we found a significantly stronger preference for the first technique used, a likely effect of *priming*.

To control specifically for effects of the experiment tasks, we also used Wilcoxon signed-ranks test to control for differences between the BANK and PEN cases (regardless of period and technique used). Again, we found no significant differences for understanding, diagram use or perception, suggesting that the two cases were comprehended and perceived as similar by the participants. However, the participants were able to identify significantly more vulnerabilities and mitigations for the BANK case than for PEN. A possible explanation is that our participants had more personal experience with the former domain.

Grizzle [89], [90] proposes a stronger test for carryover effects. For each participant and each dependent variable, Grizzle's test creates a new cumulative measure that is the sum of the measures for that participant and variable over both experiment periods. In other words, the measures for all dependent variables from the first and second period are added for each participant. A test of independent samples is then used to compare the cumulative measures of the participants according to which treatment they received first. We also conducted such as test on our data, using a Wilcoxon rank-sum test to compare the cumulative measures of groups A and B with those of C and D. We found significant differences only for the perception variables. This result confirms the above finding that priming effects may have affected our perception data (ACC_*x*) in the second experiment, but strengthens our view that carryover effects has not biased our other measures.

5. Discussion

5.1 Main results

The first experiment: The first experiment had 33 participants, divided into a MUCM group and a control group that used separate diagrams of misuse cases and of system architecture. All participants were allowed to use both the diagram and the textual description of the intrusion when solving the tasks. Participants in the MUCM group achieved significantly better understanding of the intrusion and were able to suggest significantly more mitigations. However, there were no significant overall differences in how positively the groups perceived the technique they were assigned. Correlation analyses showed that participants who understood a scenario better also identified significantly more security issues, but they were not more ready to accept the used technique. Nor did participants who identified more security issues accept their technique more readily.

The second experiment: The second experiment had 54 participants, divided into four groups that solved the experiment tasks twice, each group using the two techniques on the two tasks in a different order (Table 2). The participants were not allowed to use the textual descriptions of the intrusions when solving the tasks, and they were asked afterwards to estimate how much they had relied on the diagrams when solving the tasks. MUCMs again gave significantly better understanding of the intrusions than separate diagrams of misuse cases and of system architecture. This time, there were no significant differences for the identification task, but MUCM was perceived significantly more positively than MUC-SAD, i.e., the participants rated the technique more highly in terms of perceived usefulness, perceived ease of use, and intention to use. Correlation analyses confirmed that participants who understood a scenario better also identified significantly more security issues. In addition, both understanding a scenario better and identifying more security issues were significantly and positively correlated with perceiving the technique used more positively.

Concerning diagram use, MUCMs were used significantly more often that separate MUCs and SADs. Diagram use was also significantly correlated with both understanding of the intrusion, identification of security issues, and acceptance of the technique used.

Both experiments: Both experiments suggested that using MUCM results in significantly better *understanding* of the intrusions than the alternative representation. The difference was more significant in the second experiment, just as we would expect when the difference in treatment was sharpened by removing the textual case description while solving the tasks.

Although the MUCM users in the first experiment performed significantly better in the identification task overall and for vulnerabilities, we found no significant differences in the second one. A possible explanation is that the students might have shunned away from the more complex looking MUCM in the first experiment and tended to use the text more. Since the text is more detailed and may also be easier to understand and work with than a new notation just learnt, this approach might have been more effective. The MUC-SAD users, on the other hand, looked at diagrams with somewhat smaller complexity that did not scare them off, but not finding the information they needed there, wasted more time and thus found fewer vulnerabilities and mitigations. Another possible explanation is that MUCM is clearly, but just a little, better than MUC-SAD for identifying vulnerabilities and mitigations from scratch. In addition, MUCM is clearly, but just a little, better than MUC-SAD for identifying mitigations based on already identified vulnerabilities. Therefore, the difference between MUCM and MUC-SAD becomes clearer for mitigations than for vulnerabilities (even significantly so in the first experiment). The qualitative analysis of vulnerability types reported in [27] supports this explanation because, for each vulnerability type, the numbers of vulnerabilities and mitigations found were significantly correlated for the MUCM group, but not for the MUC-SAD group.

Why did MUCM users do better for the understanding task, but not for identification of vulnerabilities and mitigations? For understanding there was a fixed number of questions to answer, where each could be right or wrong. For vulnerabilities and mitigations, participants could identify as many as they wanted to, hence there might have been a saturation ("done enough") effect where students, who could potentially have found more if they did their best, were not optimally productive. For understanding, the same students might have been more likely to perform closer to their best when asked to respond directly to a fixed set of questions. Moreover, many students proposed common sense vulnerabilities that were fairly generic and not always closely related to the case, whereas the understanding questions were always closely related to the case, creating a stronger need to inspect the textual and diagrammatic case descriptions.

In the second experiment, but not in the first, participants rated MUCM significantly higher than MUC-SAD in terms of perceived usefulness, perceived ease of use, and intention to use. A possible explanation is that priming effects were in place in the second experiment: participants rated MUCM more highly in the second period after having been exposed to MUC-SAD in the first. Another explanation may again be the availability of textual descriptions. When textual descriptions were removed in the second experiment, and the participants were forced to resort to diagrams in the understanding task, the MUCM group might have found answers more readily in their diagrams than the MUC-SAD group in theirs. The MUC-SAD users might have become frustrated and scored their diagrams more poorly on acceptance. This explanation is supported by the participants' self-estimated diagram use: for all tasks, MUCM was significantly more used than the alternative. Correlation analysis showed that using MUCM for understanding and identification was positively correlated with all perception measures.

Correlation analyses also showed that, in both experiments, participants who understood a scenario better also identified significantly more security issues. In the second experiment, but not in the first, both understanding and identification were significantly and positively correlated with positively perceiving the technique. The difference can possibly be explained in the same way as the differences in perception between the two experiments.

Comparing the overall results of the two experiments, we see that scores for both understanding and identification are higher in the first than in the second experiment. This is no surprise, as the first experiment used students with an extra year of ICT studies behind them. In addition, having textual descriptions available the whole time may have made the first experiment simpler.

5.2 Implications for practice

Our results indicate that diagrams that combine security and architectural aspects, like MUCMs, can indeed be helpful in practical settings. Their primary use is to increasing understanding of security issues in an architectural context among people who are not ICT or security experts. MUCMs aids identification of vulnerabilities and, in particular, mitigations, at least as well as existing techniques, and may be better in some settings, particularly involving people who are not ICT or security experts. MUCMs are also likely to be at least as positively perceived as comparable techniques – sometimes even better.

Our results suggest that MUCMs may be particularly useful when they are first used for identifying vulnerabilities and then for identifying corresponding mitigations. Hence, the technique is particularly useful when used to cover both steps in sequence.

Although MUCM seems to serve its specific purpose well – that of analysing security issues in architecture contexts – other types of security analysis may not be equally well supported. MUCMs should therefore usually be used in combination with other, complementary techniques.

5.3 Implications for research

Additional work is needed to explore misuse case maps in several directions. Our two experiments have produced useful results, but we know little about how far outside our experimental setting the results are useful. A MUCM editor [91] has already been implemented, which should enable more comprehensive further validations of MUCM than the paper and pencil work reported here.

Some further validations should be bigger experiments, possibly involving practitioners instead of students. If possible, the practitioners should work on larger intrusion cases than the ones we have used here and for longer time, possibly in groups. Further experimental evaluations should also try to analyse the types and quality of threats and mitigations identified. And they should, among other things, try to resolve the conflicting results from the two experiments presented here regarding the identification task and technique acceptance. Other dependent variables may also be investigated, such as *creating diagrams from scratch*, *extending incomplete diagrams* or *using provided diagrams* for *creative problem solving* along the lines of the third experiment in [92]. Further control variables may be introduced to control for personality types (e.g., [93], [94]) or for the cultural backgrounds of participants [95], [96].

Other further validations should be case studies carried out with industrial partners. Case studies are needed to explore MUCM in detail and to develop it further and clarify its relation to other requirements, architecture and security techniques. Such case studies might either take the form of: *live cases* that use MUCMs for security analysis in real systems development or evaluation projects; *shadow cases* that use MUCMs in parallel with such projects, using the same data as inputs and involving actual project participants in the evaluation, but using established techniques instead of MUCMs in the main project; or *canned cases* that uses MUCMs to retrace finished projects, using as much as possible the same data as inputs and involving as many as the original project participants in the evaluation. Action research studies to investigate introduction of MUCMs in actual projects is another possibility, as is design research studies to improve the technique further and provide tool support.

The MUCM notation itself can possibly be improved in many ways. A particular challenge is to find the right balance between expressiveness on the one hand and ease of use and communication on the other. Already the preliminary evaluation [16] has suggested that the current notation is too complex. Further research might investigate whether simpler subsets of the notation presented here are more appropriate for less knowledgeable and experienced users, leaving the full notation for expert users. We believe that the MUCM 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 circumstance.

MUCM have been proposed to serve a specific need that, to the best of our knowledge, has not so far been well covered by existing techniques: analysis of security issues in an architecture context. Other types of security analysis may not be equally well covered, and further research is needed on whether and how to combine other techniques with MUCM, e.g., techniques for risk analysis [4] or for investigation other types of threats besides security, such as safety and privacy threats [55], [97]. Although proposals such as UMLSec [1], [2], UMLintr [68], KAOS's security extension (KAOS-SE, [3]) and Secure Tropos [5], [98] are methods that integrate several techniques to provide broader support for security work than MUCMs do, it might be interesting to investigate whether and how they could exploit MUCMs and the ideas behind them. Incorporating MUCM into our Hacker Attack Representation Method proposal [99] is another, more straightforward, avenue.

The support for understanding of intrusions with MUCM shows potential which opens up more uses than just requirements analysis. For example, it would be worth to try it in designing more complex penetration tests where the vulnerabilities of a system can be chained together to investigate their maximum impact. This way also the risk analysis can benefit from using MUCM when deciding about which vulnerability, how should be mitigated. Penetration tests are usually applied on installed systems. MUCM offers a possibility to try out some intrusion traces on the design of a system which might save time and resources compared to late penetration tests of the installed system.

5.4 Threats to validity

This section discusses the validity of our results, considering the four validity types presented in [100]: construct, internal, conclusion, and external validity.

Construct validity: Construct validity concerns whether it is legitimate to infer from the measures we made in the experiment to the theoretical constructs we were trying to answer research questions about. We have chosen dependent variables that reflected three aspects we considered central to all security analysis: *understanding* of the proposed system and of existing and potential security issues, *identification* of security issues, such as vulnerabilities and mitigations, and *acceptability* of the method, tools, and techniques used in terms of perceived usefulness, perceived ease of use, and intention to use again. *Understanding* was measured using 20 true/false statements chosen to reflect typical security issues that must be understood in security analysis. *Identification* was measured by asking participants to list as many vulnerabilities and corresponding mitigations as possible. Both understanding and identification were measured using real intrusions from the literature. Of course, there are other ways to explore understanding and identification, but the ability to answer questions about a case should be a reasonable approximation of understanding, and the identification of vulnerabilities and mitigations are central tasks in secure software engineering. *Acceptability* was measured using questions adapted from a thoroughly validated, well known and widely used instrument for user acceptance of technology, TAM [83]. We thus argue that the three chosen measures – understanding, identification, and acceptability – are all highly relevant and suitably operationalised in our experiment, although we readily admit that we could have included other measures as well, such as *model construction*, *model enhancement/extension* and *creative problem solving*.

We have measured identification effectiveness directly by *counting* the numbers of vulnerabilities and mitigations identified by each participant, without taking the *quality* of the vulnerabilities and mitigations into account. These direct measures are well suited for assessing the usefulness of MUCMs in the generative or creative stages during early RE, where a central task is to look at the proposed architecture from many different perspectives in order to find as many potential security threats and possible solutions as possible, and where even bad or irrelevant suggestions may be welcome because they may aid generating further ideas. Our direct measures are less well suited for assessing how well MUCMs support identifying the most important vulnerabilities and most promising mitigations for a given problem. To take quality of vulnerabilities and mitigations into account, further work should introduce more specific measures, either by counting vulnerabilities and mitigations that are deemed acceptable through an explicit procedure or by weighing each vulnerability or mitigation through an objective quality assessment.

In addition to our chosen focus on vulnerabilities and mitigations in the identification task, it would have been possible to use identification of threats ("IDEN_THR") as a third measure, alongside IDEN_VUL and IDEN_MIT, because a vulnerability can be targeted by several threats and a mitigation is a direct countermeasure to one or more of these threats (more directly than it is a countermeasure to some vulnerability that the threats target). We nevertheless did not consider threats in our experiments because it would have made the experiment tasks even larger, taking up limited experiment time. Also, because vulnerabilities may appear similar to threats for the non-expert, asking our student participants to identify both might have created confusion. Investigating identification of threats in addition to vulnerabilities and mitigations remains an interesting possibility for future studies that involve more experienced subjects.

Internal validity: Internal validity assesses whether the observed outcomes were due to our treatment or to other factors. We will separately discuss our *experiments groups*, our *treatments* and, for the second experiment, potential *carryover and related effects*.

Experiment groups: One threat to internal validity is *unbalanced experiment groups*. In both experiments, we therefore assigned participants randomly to groups. We also administered a background questionnaire to control for confounding subjective factors, and we used statistical testing to control for differences between groups. We only found two significant differences:

- In the first experiment, the participants in the MUCM group reported significantly longer work experience than the other group. This was due to two MUCM-subjects who reported 20 and 36 years of ICT-related work experience. Because the instructor cannot remember any older students attending the class, we strongly suspect they misreported their experience in months rather than years.
- In the second experiment, the two *MUCM first* groups A and B (with means of 1,5 and 0,6 respectively) reported significantly shorter work experience than the two *MUCM last* groups C and D (means 3,1 and 2,8). (Misreporting is again a possible reason for the difference.) Although our crossover design should cancel out these differences for the most part, they may have interacted with carryover effects. We therefore did another Spearman correlation analysis between ICT-work experience and all the other main

variables (with data from both periods pooled), but found no significant correlations. We thus conclude that differences in ICT work experience between the groups are unlikely to have influenced our results.

Treatments: Another threat to internal validity is the *treatments* we have used. We made sure each group performed the same experiment tasks in the same auditorium, subjected 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, diffusion of treatments and interaction between participants was unlikely to be a problem.

Another internal validity threat related to treatments is that the MUCM notation has been proposed by the authors and that this may – perhaps inadvertently – have introduced bias in the training of participants. We have tried to avoid this by designing the tutorials for the two treatments to be as similar as possible. We have also attempted to control for it by having the students self-report their knowledge about MUCMs before the experiment, showing that they had little such knowledge. Finally, we note that the authors have not only proposed the MUCM treatment, but also the MUC notation used in the alternative treatment.

A final internal validity threat related to treatments is that the MUCM and MUC-SAD treatments have provided the participants with different information about the intrusion cases. Hence, instead of comparing different ways of presenting the same information – which was the intention of our experiments – we have also compared availability of different information. According to [101], [102], two representations are *informationally equivalent* if the transformation from one to the other entails no loss of information, i.e., if each can be constructed from the other. We carefully designed the MUCM, MUC, and SAD diagrams to approach informational equivalence, but our MUCM diagrams nevertheless contained information available in neither the MUC nor the SAD diagrams, namely how a **Vulnerability** relates to a **Concrete Element** and how a **Malicious Action** is made up of several **Attacks**. For the MUC-SAD users, this information was only available in the textual description. At the same time, our MUC diagrams showed information not available in the MUCM, such as the **Attacker** and the **Affected Element** (especially the **Function**). The MUCM diagrams did not show this regular functionality alongside the security threats. We suspect the impact of these differences on task performance would depend on the nature of the experiment tasks: for identification tasks, it is not surprising that MUCM is most effective whereas, for questions about types of attackers, functions affected by the attacks, etc., we might have expected MUC-SAD to provide better understanding. Although our treatments were thus not informationally equivalent, we argue they were the closest informational match that was possible to find by simple combination of existing, well-known techniques that are used in practice. Introducing new or complex combinations of techniques as our alternative treatment might have increased internal validity, but would have reduced external validity and practical relevance more than correspondingly.

Carryover and related effects: In the second experiment, we also used a crossover or Latin-Squares experimental design. This design contributed to eliminating potential problems with subjective factors learning effects, boredom and fatigue, but it also introduced validity issues of its own, which we have already discussed in Section 4.8.

Conclusion validity: Conclusion validity concerns the statistical relationship between our treatments and our results: whether we accepted or rejected our hypotheses. Is our sample size sufficient to justify our conclusions? A Type I error is to incorrectly reject a true null hypothesis, i.e., to conclude that there is a difference between MUCM and MUC-SAD for a dependent variable when there really is none. A Type II error is to incorrectly accept a false null hypothesis, i.e., to conclude that there is no difference between MUCM and MUC-SAD for a dependent variable when there really is one.

We let the acceptable probability Type I error be α=0.05 (our required significance level) and the acceptable probability of a Type II error be β=0.2, so that 1-β=0.8 is the required power (or sensitivity) of our tests. For each test, Tables 7 and 9 show the sample sizes to required to justify our conclusion given the observed effect as well as the effect required to justify our sample sizes. We have calculate the effects and sample sizes using the pooled standard deviations of each group of participants and using actual differences between means instead of minimal detectable differences. (We note that the required sample and effect sizes of Tables 7 and 9 must be used with caution, because they used a Student t-test and our data were not in general normally distributed.)

Concerning Type I errors – finding a relationship that was not really present due to a too small sample size – Table 7 shows that, in the first experiment, our sample size was only large enough to justify a positive conclusion about UND_ALL, although it came close for the others (UND, IDEN_MIT, and IDEN) too. In the second experiment, with a Latin-Squares arrangement and more participants, Table 9 shows that our sample size was indeed sufficient to support all our positive conclusions (about UNT_TOT, UND_ALL, ACC_x, DIAG_UND, and DIAG_VUL), but not about DIAG_MIT, for which it however came close (d=0.36 shown versus d=0.39 required). We conclude that Type I error is potentially an issue with conclusions about UND, but not with the closely related measure UND_ALL. Type I error is also potentially an issue with conclusions about DIAG_MIT, but not with the closely related DIAG_UND and DIAG_VUL.

Concerning Type II errors – missing a relationship that was really present due to a too small sample size – we note that we have only rejected hypotheses for variables (IDEN_*x*) for which the effect was not even in the right direction in the second experiment, meaning that sample size did not play a role. Effect analysis also lends additional justification to not rejecting hypothesis H3, for which we had sufficiently large samples for all the significant differences (for ACC_PU, ACC_ITU, and ACC) in the second experiment.

External validity is concerned with generalisation from the experimental setting to other situations; in our case, most importantly to industrial systems development. This was central for the setup of both our experiments.

Experiment participants may threaten external validity if their behaviours differ from typical participants in industrial security assessments. Although our participants were not industrial practitioners, our student participants had already taken several semesters of ICT courses and, as observed in [103], level of competency is a better predictor of experiment performance than whether someone is a student or a practitioner. Also, although our 2nd and 3rd year student participants had somewhat lower competence than a junior consultant who would enter the ICT workforce after a 3-year Bachelor's degree, they most likely had higher ICT competence than the security and other non-ICT experts who are also involved in security assessments. In terms of ICT competence, our mix of students from different semesters may thus be comparable to the mix of professionals typically involved in security assessments.

Moreover, because the experiment involved a new technique, the difference between students and experienced practitioners is likely to be smaller than if we only used methods that were well-known to practitioners. As shown in [104], [105], students can be expected to be quite representative of practitioners when trying out a new technique for the very first time. [106] compared how groups of freshmen, graduate students, and practitioners used an ICT technique and, although there were differences between freshmen and graduate students and, though less so, between graduate students and practitioners, all three groups showed the same preference for and better performance with one technique over the other. Hence, using students in the first experiments with a new technique makes sense: it provides useful indications of the relative strengths of the new technique and (as suggested in [107]) serves to validate the experiment design for future experiments with practitioners.

Duration and task size are central threats to external validity. A controlled experiment that lasts for a couple of hours can never be fully representative of huge and complex problems encountered in real development situations. We have tried to remedy this in part by using realistic attack scenarios written by an experienced white-hat hacker, and we have given participants typical problems to solve, such as understanding the case and identifying vulnerabilities and mitigations. Our conclusions should thus at least be generalisable to working with single scenarios, although they may not be generalisable to assessing the security of entire systems. Such a complex task would in any case be hard to investigate experimentally, instead pointing towards future industrial case studies that will allow us to observe use of MUCMs in more realistic and larger project settings.

As for duration, the time spent learning and using each technique in our experiments was limited, which may favour techniques that are easy to understand quickly and effective for identifying and expressing easy-to-find threats, such as variations of standard security threats that are already described in the security literature, as opposed to problem- or domain-specific threats. Only further experiments can reliably assess the impact of limited duration and task size on our results.

Participant motivation is another external validity issue. People involved in real security requirements work have natural incentives to perform well, such as the wish to do a good job, avoiding security attacks, making a good impression, earning a bonus, etc. Our student participants did not have similar natural incentives but, as ICT students, they had already chosen to invest several years of their lives to learn about ICT. Having volunteered to participate in the experiment for its full duration, they were thus likely to dedicate a reasonable effort to the experiment tasks in order to maximise learning about two timely and important topics: modelling and security.

Control treatment may affect external validity in addition to internal validity, which we already discussed. We think our choice of MUCs combined with SADs stroke a good balance between realism and control. There are of course many other possible ways of representing and assessing ICT security issues (e.g., [1]–[3], [5], [68], [98]) and of displaying system architecture (e.g., [11], [13], [76], [77]), but none of them are good candidates for experimental comparisons with MUCMs because none of them cover both security and architecture aspects to the same degree as MUCMs. The combination of MUC and SAD, on the other hand, increases external validity by using standard notations used in industry.

Textual descriptions were not made available to the participants of the second experiment while they solved their tasks. This increased internal validity, but possibly at the cost of decreasing external validity. In practical security analysis, which may take a long time, analysts are of course free to use the materials they like whenever they like during analysis, whether textual or not. The second experiment may therefore have been less generalisable to this type of prolonged security work than the first. But in practical security analyses, stakeholders are also likely to spend shorter intervals focussing only on specific materials one at a time, such as one or more diagrams. The second experiment may be generalisable to this more specific episodic part of security work. Both types of knowledge may be useful, although it seems that experiments that involve extensive tasks to be solved over longer time should attempt to control for, rather than restrict, use of different types of materials.

6. Conclusions and Further Work

The paper has presented the first thorough experimental evaluation of misuse case maps, a new technique that combines ideas from misuse cases for security threat modelling with use case maps for scenario and systems architecture modelling. The idea is to better support security requirements work in an architectural context during early requirements work. This idea is in line with [108], who includes architectural risk analysis as a central *touchpoint* between software security management and the general software development process [109], to which we add that architecture is not only important during the architecture and design stages, but from the earliest requirement stages on.

Our results indicate that misuse case maps are indeed an improvement over combined use of separate misuse case and system architecture diagrams. In particular, MUCM users showed significantly better understanding of the intrusion scenarios they were presented with, MUCM users were able to identify significantly more mitigations to identified threats, and MUCM diagrams were more used than separate MUC and SAD diagrams in combination. In the second experiment and overall, MUCM was perceived significantly more positively by its users.

Our experiments thus suggest that misuse case maps are indeed a useful addition to the arsenal of modelling techniques that developers and security experts have at their disposal while collaborating to develop secure software. MUCMs are useful for making intrusions understood by people who are not security experts and for identifying ways to mitigate threats. They are also useful because they may be acceptable for people who are not security experts. As pointed out in [63], there are already many modelling techniques available, so one must always be careful when proposing a new one. Nevertheless, different problems need different techniques [110], and new modelling approaches are sometimes justified. There is a particular need for covering modelling perspectives that have not previously been given enough attention. We argue that security requirements work in an architectural context during early requirements work is such a perspective.

Acknowledgement

This work was part of the ReqSec project funded by the Norwegian Research Council.

References

[1] J. Jürjens, *Secure systems development with UML*. Springer, 2005.

[2] J. Jürjens, “UMLsec: Extending UML for secure systems development,” in *≪ UML≫ 2002—The Unified Modeling Language*, Springer, 2002, pp. 412–425.

[3] A. Van Lamsweerde, “Elaborating security requirements by construction of intentional anti-models,” presented at the Proceedings of the 26th International Conference on Software Engineering, 2004, pp. 148–157.

[4] T. Dimitrakos, B. Ritchie, D. Raptis, J. Ø. Aagedal, F. den Braber, K. Stølen, and S. H. Houmb, “Integrating Model-based Security Risk Management into eBusiness Systems Development,” in *Towards the Knowledge Society*, Springer, 2003, pp. 159–175.

[5] P. Giorgini, F. Massacci, J. Mylopoulos, and N. Zannone, “Modeling security requirements through ownership, permission and delegation,” presented at the Requirements Engineering, 2005. Proceedings. 13th IEEE International Conference on, 2005, pp. 167–176.

[6] J. McDermott and C. Fox, “Using abuse case models for security requirements analysis,” presented at the Proc. 15th Annual Computer Security Applications Conference (ACSAC’99), 1999, pp. 55–64.

[7] G. Sindre and A. L. Opdahl, “Eliciting security requirements with misuse cases,” in *Proc. TOOLS-PACIFIC’00*, 2000, pp. 120–131.

[8] D. G. Firesmith, “Security use cases,” *Journal of object technology*, vol. 2, no. 3, pp. 53–64, 2003.

[9] L. Liu, E. Yu, and J. Mylopoulos, “Security and privacy requirements analysis within a social setting,” in *Proc. 11th IEEE International Requirements Engineering Conference (RE’2003)*, 2003, pp. 151–161.

[10] L. Lin, B. Nuseibeh, D. Ince, and M. Jackson, “Using abuse frames to bound the scope of security problems,” in *Proc. 12th IEEE International Requirements Engineering Conference (RE’2004)*, 2004, pp. 354–355.

[11] C. Jensen, “Secure software architectures,” presented at the Proceedings of the Eighth Nordic Workshop on Programming Environment Research (NWPER’98), 1998, pp. 239–246.

[12] Y. Deng, J. Wang, J. J. Tsai, and K. Beznosov, “An approach for modeling and analysis of security system architectures,” *Knowledge and Data Engineering, IEEE Transactions on*, vol. 15, no. 5, pp. 1099–1119, 2003.

[13] Y. Ali, S. El-Kassas, and M. Mahmoud, “A rigorous methodology for security architecture modeling and verification,” presented at the System Sciences, 2009. HICSS’09. 42nd Hawaii International Conference on, 2009, pp. 1–10.

[14] J. Wang, X. He, and 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.

[15] A. Herrmann, A. Morali, S. Etalle, and R. Wieringa, “RiskREP: Risk-based Security Requirements Elicitation and Prioritization,” in *Proc. 1st International Workshop on Alignment of Business Process and Security Modelling (ABPSM’2011)*, Riga/Latvia, 2011.

[16] P. Karpati, G. Sindre, and A. L. Opdahl, “Visualizing cyber attacks with misuse case maps,” in *Requirements Engineering: Foundation for Software Quality*, Springer, 2010, pp. 262–275.

[17] M. Jarke, P. Loucopoulos, K. Lyytinen, J. Mylopoulos, and W. Robinson, “The brave new world of design requirements,” *Information Systems*, vol. 36, no. 7, pp. 992–1008, 2011.

[18] B. Nuseibeh, “Weaving together requirements and architectures,” *Computer*, vol. 34, no. 3, pp. 115–119, 2001.

[19] N. Mayer, A. Rifaut, and E. Dubois, “Towards a risk-based security requirements engineering framework,” presented at the Workshop on Requirements Engineering for Software Quality. In Proc. of REFSQ, 2005, vol. 5.

[20] M. Jarke, R. Klamma, K. Pohl, and E. Sikora, “Requirements engineering in complex domains,” in *Graph transformations and model-driven engineering*, Springer, 2010, pp. 602–620.

[21] V. Katta, P. Karpati, A. L. Opdahl, C. Raspotnig, and G. Sindre, “Comparing two techniques for intrusion visualization,” in *The Practice of Enterprise Modeling*, Springer, 2010, pp. 1–15.

[22] G. Sindre and A. L. Opdahl, “Eliciting security requirements with misuse cases,” *Requirements Engineering*, vol. 10, no. 1, pp. 34–44, 2005.

[23] R. J. Buhr, “Use case maps for attributing behaviour to system architecture,” presented at the Parallel and Distributed Real-Time Systems, 1996. Proceedings of the 4th International Workshop on, 1996, pp. 3–10.

[24] R. J. Buhr and R. S. Casselman, *Use case maps for object-oriented systems*, vol. 302. Prentice Hall Englewood Cliffs, 1996.

[25] D. Amyot, “Use Case Maps Quick Tutorial,” *See http://www. usecasemaps. org/pub/UCMtutorial/UCMtutorial. pdf*, 1999.

[26] P. Karpati, A. L. Opdahl, and G. Sindre, “Experimental evaluation of misuse case maps for eliciting security requirements,” presented at the Proc. 1st Security Conference–Europe (ISCE’2010), Örebro/Sweden, 2010.

[27] P. Karpati, A. L. Opdahl, and G. Sindre, “Experimental Comparison of Misuse Case Maps with Misuse Cases and System Architecture Diagrams for Eliciting Security Vulnerabilities and Mitigations,” presented at the Sixth International Conference on Availability, Reliability and Security (ARES’2011), 2011, pp. 507–514.

[28] I. A. Tøndel, M. G. Jaatun, and P. H. Meland, “Security requirements for the rest of us: A survey,” *Software, IEEE*, vol. 25, no. 1, pp. 20–27, 2008.

[29] D. Mellado, C. Blanco, L. E. Sánchez, and E. Fernández-Medina, “A systematic review of security requirements engineering,” *Computer Standards & Interfaces*, vol. 32, no. 4, pp. 153–165, 2010.

[30] B. Fabian, S. Gürses, M. Heisel, T. Santen, and H. Schmidt, “A comparison of security requirements engineering methods,” *Requirements engineering*, vol. 15, no. 1, pp. 7–40, 2010.

[31] P. Karpati, G. Sindre, and A. L. Opdahl, “Characterising and analysing security requirements modelling initiatives,” presented at the Availability, Reliability and Security (ARES), 2011 Sixth International Conference on, 2011, pp. 710–715.

[32] P. Salini and S. Kanmani, “Survey and analysis on security requirements engineering,” *Computers & Electrical Engineering*, vol. 38, no. 6, pp. 1785–1797, 2012.

[33] D. Muñante, V. Chiprianov, L. Gallon, and P. Aniorté, “A Review of Security Requirements Engineering Methods with Respect to Risk Analysis and Model-Driven Engineering,” in *Availability, Reliability, and Security in Information Systems*, Springer, 2014, pp. 79–93.

[34] A. V. Uzunov, E. B. Fernandez, and K. Falkner, “Engineering Security into Distributed Systems: A Survey of Methodologies.,” *J. UCS*, vol. 18, no. 20, pp. 2920–3006, 2012.

[35] L. Lucio, Q. Zhang, P. H. Nguyen, M. Amrani, J. Klein, H. Vangheluwe, and Y. Le Traon, “Advances in Model-Driven Security.,” *Advances in Computers*, vol. 93, pp. 103–152, 2014.

[36] I. Jacobson, *Object-oriented software engineering: a use case driven approach*. Pearson, 1992.

[37] G. Elahi, E. Yu, and N. Zannone, “A vulnerability-centric requirements engineering framework: analyzing security attacks, countermeasures, and requirements based on vulnerabilities,” *Requirements engineering*, vol. 15, no. 1, pp. 41–62, 2010.

[38] G. Elahi, “Making Trade-offs among Security and Other Requirements during System Design,” Ph.D. Dissertation, University of Toronto, Toronto/Canada, 2012.

[39] A. Rodríguez, E. Fernández-Medina, and M. Piattini, “Towards an integration of Security Requirements into Business Process Modeling,” in *Security in Information Systems, Proceedings of the 3rd International Workshop on Security in Information Systems (WOSIS 2005), in conjunction with ICEIS2005*, Miami/USA, 2005, pp. 287–297.

[40] G. Sindre, “Mal-activity diagrams for capturing attacks on business processes,” in *Requirements Engineering: Foundation for Software Quality*, Springer, 2007, pp. 355–366.

[41] A. Rodríguez, E. Fernández-Medina, and M. Piattini, “Capturing security requirements in business processes through a UML 2.0 activity diagrams profile,” in *Advances in Conceptual Modeling-Theory and Practice*, Springer, 2006, pp. 32–42.

[42] E. G. Amoroso, *Fundamentals of computer security technology*. Prentice-Hall, Inc., 1994.

[43] B. Schneier, *Secrets and lies: digital security in a networked world*. Wiley. com, 2011.

[44] S. Bistarelli, F. Fioravanti, and P. Peretti, “Defense trees for economic evaluation of security investments,” presented at the Availability, Reliability and Security, 2006. ARES 2006. The First International Conference on, 2006, p. 8–pp.

[45] K. S. Edge, G. Dalton, R. A. Raines, and R. F. Mills, “Using attack and protection trees to analyze threats and defenses to homeland security,” presented at the Military Communications Conference, 2006. MILCOM 2006. IEEE, 2006, pp. 1–7.

[46] S. A. Zonouz, H. Khurana, W. H. Sanders, and T. M. Yardley, “RRE: A game-theoretic intrusion Response and Recovery Engine,” presented at the Dependable Systems & Networks, 2009. DSN’09. IEEE/IFIP International Conference on, 2009, pp. 439–448.

[47] A. Roy, D. S. Kim, and K. S. Trivedi, “Cyber security analysis using attack countermeasure trees,” presented at the Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research, 2010, p. 28.

[48] J. Wang, J. N. Whitley, R. C.-W. Phan, and D. J. Parish, “Unified parametrizable attack tree,” *International Journal for Information Security Research*, vol. 1, no. 1, pp. 20–26, 2011.

[49] T. Boswell, “Specification and validation of a security policy model,” presented at the FME’93: Industrial-Strength Formal Methods, 1993, pp. 42–51.

[50] A. Hall and R. Chapman, “Correctness by construction: Developing a commercial secure system,” *Software, IEEE*, vol. 19, no. 1, pp. 18–25, 2002.

[51] “Common criteria for information technology security evaluation,” National Security Agency, Common Criterian Implementation Board, Technical Report CCIMB-99-031, 2002.

[52] D. G. Firesmith, “A taxonomy of security-related requirements,” presented at the International Workshop on High Assurance Systems (RHAS’05), 2005.

[53] F. Massacci, J. Mylopoulos, F. Paci, T. T. Tun, and Y. Yu, “An extended ontology for security requirements,” in *Advanced Information Systems Engineering Workshops*, London/UK, 2011, pp. 622–636.

[54] A. Gangemi, N. Guarino, C. Masolo, A. Oltramari, and L. Schneider, “Sweetening ontologies with DOLCE,” in *Knowledge engineering and knowledge management: Ontologies and the semantic Web*, Springer, 2002, pp. 166–181.

[55] C. Raspotnig and A. Opdahl, “Comparing risk identification techniques for safety and security requirements,” *Journal of Systems and Software*, vol. 86, no. 4, pp. 1124–1151, 2013.

[56] N. Mayer, P. Heymans, and R. Matulevicius, “Design of a Modelling Language for Information System Security Risk Management.,” presented at the RCIS, 2007, pp. 121–132.

[57] R. Matulevicius, N. Mayer, and P. Heymans, “Alignment of misuse cases with security risk management,” presented at the Availability, Reliability and Security, 2008. ARES 08. Third International Conference on, 2008, pp. 1397–1404.

[58] É. Dubois, P. Heymans, N. Mayer, and R. Matulevičius, “A systematic approach to define the domain of information system security risk management,” in *Intentional Perspectives on Information Systems Engineering*, Springer, 2010, pp. 289–306.

[59] M. J. M. Chowdhury, R. Matulevičius, G. Sindre, and P. Karpati, “Aligning mal-activity diagrams and security risk management for security requirements definitions,” in *Requirements Engineering: Foundation for Software Quality*, Springer, 2012, pp. 132–139.

[60] C. Raspotnig, P. Karpati, and V. Katta, “A Combined Process for Elicitation and Analysis of Safety and Security Requirements,” in *Enterprise, Business-Process and Information Systems Modeling*, Springer, 2012, pp. 347–361.

[61] K. Labunets, F. Massacci, F. Paci, and L. M. S. Tran, “An experimental comparison of two risk-based security methods,” presented at the Empirical Software Engineering and Measurement, 2013 ACM/IEEE International Symposium on, 2013, pp. 163–172.

[62] F. Massacci and F. Paci, “How to select a security requirements method? a comparative study with students and practitioners,” in *Secure IT Systems*, Springer, 2012, pp. 89–104.

[63] A. L. Opdahl and G. Sindre, “Experimental comparison of attack trees and misuse cases for security threat identification,” *Information and Software Technology*, vol. 51, no. 5, pp. 916–932, 2009.

[64] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos, “Tropos: An agent-oriented software development methodology,” *Autonomous Agents and Multi-Agent Systems*, vol. 8, no. 3, pp. 203–236, 2004.

[65] R. Matulevičius, N. Mayer, H. Mouratidis, E. Dubois, P. Heymans, and N. Genon, “Adapting secure tropos for security risk management in the early phases of information systems development,” in *Advanced Information Systems Engineering*, 2008, pp. 541–555.

[66] A. Dardenne, A. Van Lamsweerde, and S. Fickas, “Goal-directed requirements acquisition,” *Science of computer programming*, vol. 20, no. 1, pp. 3–50, 1993.

[67] D. Basin, J. Doser, and T. Lodderstedt, “Model driven security: From UML models to access control infrastructures,” *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 15, no. 1, pp. 39–91, 2006.

[68] M. Hussein and M. Zulkernine, “UMLintr: a UML profile for specifying intrusions,” in *13th Annual IEEE International Symposium and Workshop on Engineering of Computer Based Systems (ECBS 2006)*, Potsdam/Germany, 2006, pp. 279–288.

[69] S. H. Houmb, S. Islam, E. Knauss, J. Jürjens, and K. Schneider, “Eliciting security requirements and tracing them to design: an integration of Common Criteria, heuristics, and UMLsec,” *Requirements Engineering*, vol. 15, no. 1, pp. 63–93, 2010.

[70] H. Mouratidis and J. Jürjens, “From goal-driven security requirements engineering to secure design,” *International Journal of Intelligent Systems*, vol. 25, no. 8, pp. 813–840, 2010.

[71] H. Schmidt and J. Jürjens, “Connecting security requirements analysis and secure design using patterns and UMLsec,” presented at the Advanced Information Systems Engineering, 2011, pp. 367–382.

[72] Object Management Group (OMG), “Unified Modeling Language.” [Online]. Available: http://www.uml.org,. [Accessed: 04-Jun-2010].

[73] M. Schumacher, E. Fernandez-Buglioni, D. Hybertson, F. Buschmann, and P. Sommerlad, *Security Patterns: Integrating security and systems engineering*, vol. 7. John Wiley & Sons, 2006.

[74] L. Liu and E. Yu, “From requirements to architectural design-using goals and scenarios,” presented at the ICSE-2001 Workshop: From Software Requirements to Architectures (STRAW’2001), 2001, pp. 22–30.

[75] M. Howard and D. LeBlanc, *Writing secure code*. O’Reilly, 2009.

[76] J. J. Pauli and D. Xu, “Misuse case-based design and analysis of secure software architecture,” presented at the Information Technology: Coding and Computing, 2005. ITCC 2005. International Conference on, 2005, vol. 2, pp. 398–403.

[77] D. Xu and J. Pauli, “Threat-driven design and analysis of secure software architectures,” *Journal of Information Assurance and Security*, vol. 1, no. 3, pp. 171–180, 2006.

[78] E. A. Oladimeji, S. Supakkul, and L. Chung, “A Model-driven Approach to Architecting Secure Software.,” presented at the SEKE, 2007, p. 535.

[79] J. C. Pelaez, E. B. Fernandez, and M. M. Larrondo-Petrie, “Misuse patterns in VoIP,” *Security and Communication Networks*, vol. 2, no. 6, pp. 635–653, 2009.

[80] K. D. Mitnick and W. L. Simon, *The Art of Intrusion: The real stories behind the exploits of hackers, intruders and deceivers*. Wiley, 2005.

[81] G. Elahi, E. Yu, and N. Zannone, “A modeling ontology for integrating vulnerabilities into security requirements conceptual foundations,” in *Conceptual Modeling-ER 2009*, Springer, 2009, pp. 99–114.

[82] D. Moody, “The ‘physics’ of notations: toward a scientific basis for constructing visual notations in software engineering,” *Software Engineering, IEEE Transactions on*, vol. 35, no. 6, pp. 756–779, 2009.

[83] F. D. Davis, R. P. Bagozzi, and P. R. Warshaw, “User acceptance of computer technology: a comparison of two theoretical models,” *Management science*, vol. 35, no. 8, pp. 982–1003, 1989.

[84] V. Venkatesh, “Determinants of perceived ease of use: Integrating control, intrinsic motivation, and emotion into the technology acceptance model,” *Information systems research*, vol. 11, no. 4, pp. 342–365, 2000.

[85] V. Venkatesh, M. G. Morris, G. B. Davis, and F. D. Davis, “User acceptance of information technology: Toward a unified view.,” *MIS quarterly*, vol. 27, no. 3, 2003.

[86] J. Cohen, *Statistical power analysis for the behavioral sciencies*, 2nd Edition. Routledge, 1988.

[87] W. G. Hopkins, “A new view of statistics,” University of Queensland, Brisbane, 2001.

[88] D. Boud, R. Keogh, and D. Walker, *Reflection: Turning experience into learning*. Routledge, 2013.

[89] J. E. Grizzle, “The two-period change-over design and its use in clinical trials,” *Biometrics*, pp. 467–480, 1965.

[90] D. Shen and Z. Lu, “Estimate carryover effect in clinical trial crossover designs,” in *Proc. Annual Conference of PharmaSUG*, 2006, p. Paper PO16.

[91] Y. Bizhanzadeh and P. Karpati, “jMUCMNav: an Editor for Misuse Case Maps,” presented at the First Int. Workshop on Alignment of Business Process and Security Modelling (ABPSM 2011), Riga, Latvia, 2011.

[92] F. Bodart, A. Patel, M. Sim, and R. Weber, “Should optional properties be used in conceptual modelling? A theory and three empirical tests,” *Information Systems Research*, vol. 12, no. 4, pp. 384–405, 2001.

[93] M. H. McCaulley, *Jung’s theory of psychological types and the Myers-Briggs Type Indicator*. Center for applications of Psychological Type, 1981.

[94] G. Saucier, “Mini-markers: A brief version of Goldberg’s unipolar Big-Five markers,” *Journal of personality assessment*, vol. 63, no. 3, pp. 506–516, 1994.

[95] G. Hofstede, *Culture’s consequences: International differences in work-related values*, vol. 5. sage, 1984.

[96] F. Trompenaars and C. Hampden-Turner, *Riding the waves of culture*. McGraw-Hill New York, 1998.

[97] M. Deng, K. Wuyts, R. Scandariato, B. Preneel, and W. Joosen, “A privacy threat analysis framework: supporting the elicitation and fulfillment of privacy requirements,” *Requirements Engineering*, vol. 16, no. 1, pp. 3–32, 2011.

[98] H. Mouratidis and P. Giorgini, “Secure tropos: A security-oriented extension of the tropos methodology,” *International Journal of Software Engineering and Knowledge Engineering*, vol. 17, no. 02, pp. 285–309, 2007.

[99] P. Karpati, A. L. Opdahl, and G. Sindre, “HARM: Hacker Attack Representation Method,” in *Software and Data Technologies*, Springer, 2013, pp. 156–175.

[100] C. Wohlin and M. Höst, “Experimentation in software engineering: An introduction,” *The Kluwer International Series In Software Engineering*, 2000.

[101] H. A. Simon, “On the forms of mental representation,” *Perception and cognition: Issues in the foundations of psychology*, vol. 9, pp. 3–18, 1978.

[102] J. H. Larkin and H. A. Simon, “Why a diagram is (sometimes) worth ten thousand words,” *Cognitive science*, vol. 11, no. 1, pp. 65–100, 1987.

[103] E. Arisholm and D. I. Sjøberg, “Evaluating the effect of a delegated versus centralized control style on the maintainability of object-oriented software,” *Software Engineering, IEEE Transactions on*, vol. 30, no. 8, pp. 521–534, 2004.

[104] M. Höst, B. Regnell, and C. Wohlin, “Using students as subjects—a comparative study of students and professionals in lead-time impact assessment,” *Empirical Software Engineering*, vol. 5, no. 3, pp. 201–214, 2000.

[105] M. Svahnberg, A. Aurum, and C. Wohlin, “Using students as subjects-an empirical evaluation,” presented at the Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement, 2008, pp. 288–290.

[106] P. Runeson, “Using students as experiment subjects–an analysis on graduate and freshmen student data,” presented at the Proceedings of the 7th International Conference on Empirical Assessment in Software Engineering.–Keele University, UK, 2003, pp. 95–102.

[107] J. C. Carver, L. Jaccheri, S. Morasca, and F. Shull, “A checklist for integrating student empirical studies with research and teaching goals,” *Empirical Software Engineering*, vol. 15, no. 1, pp. 35–59, 2010.

[108] G. McGraw, “Software Security Touchpoint: Architectural Risk Analysis,” 2009.

[109] G. McGraw, “The 7 Touchpoints of Secure Software.” Dr. Dobb’s Journal, 01-Sep-2005.

[110] D. Benyon and S. Skidmore, “Towards a tool kit for the systems analyst,” *The Computer Journal*, vol. 30, no. 1, pp. 2–7, 1987.

((Missing table “The bank hack case”, taken from ref. [80], chapter 7.))

Table *1*: Stepwise description of an intrusion into a banking computer system [80], chapter 7, as reported by the perpetrator, who claims to be white-hat hacker who misused none of the information retrieved and otherwise caused no harm to the system.

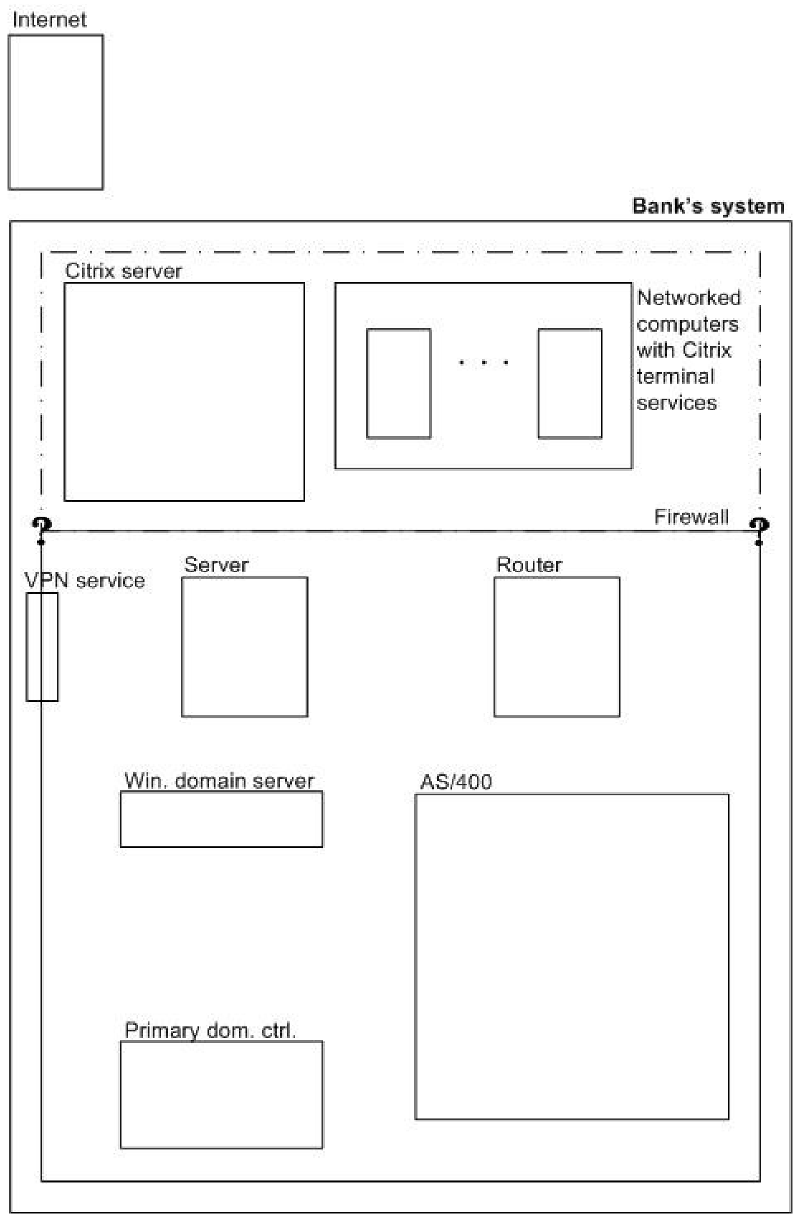


Figure 1: SAD for the bank hack case used in the experiments.

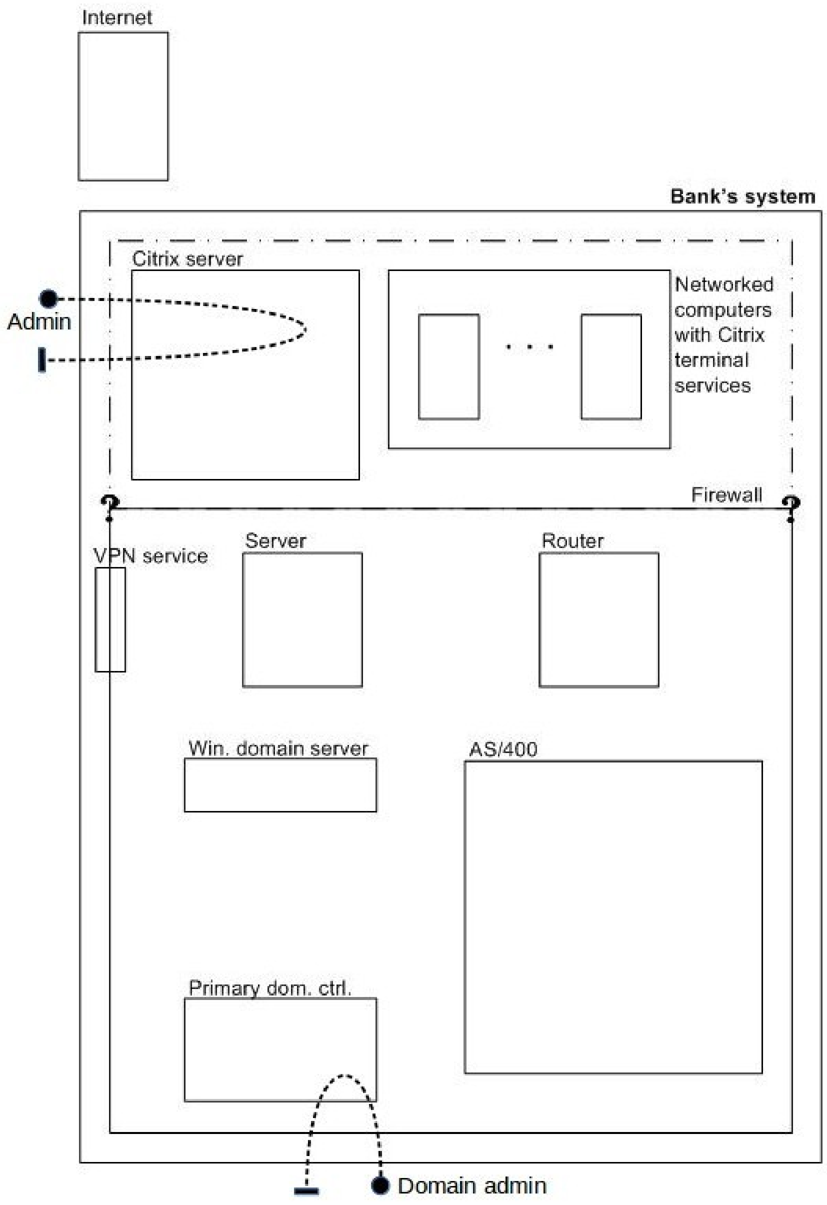


Figure 2. UCM for the bank hack case used in the experiments.

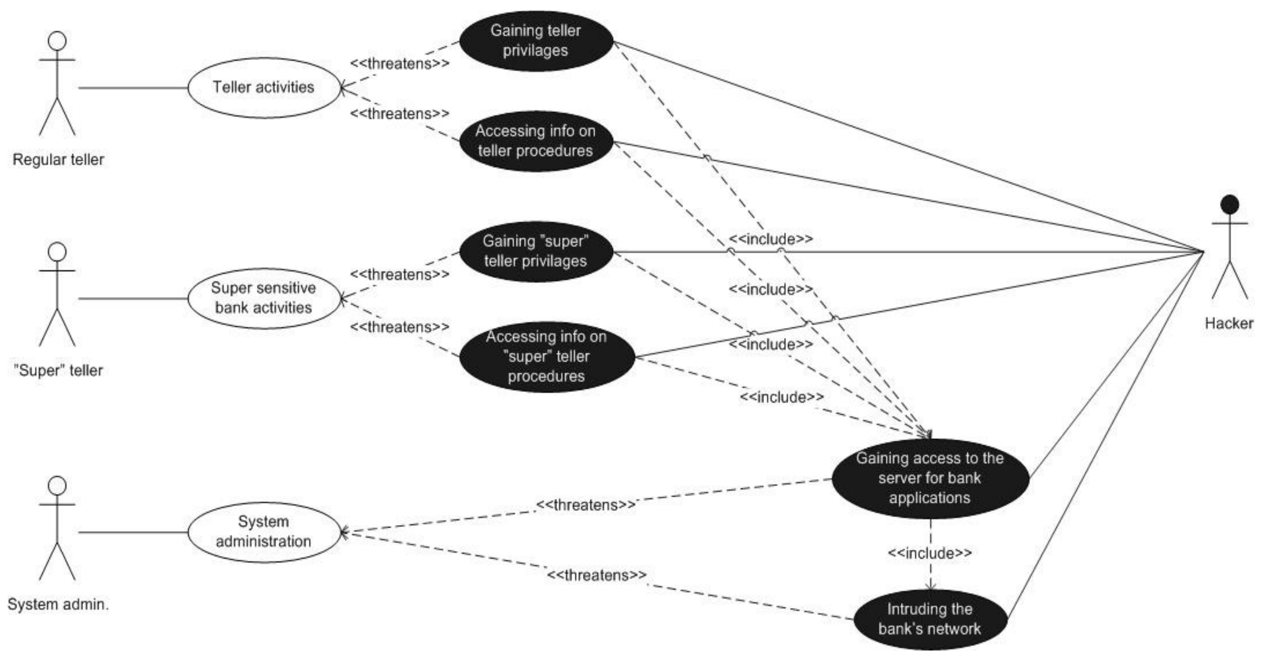

Regular teller
Teller activities
<<threatens>>
<<threatens>>
Gaining teller privilages
Accessing info on teller procedures
"Super" teller
Super sensitive bank activities
<<threatens>>
<<threatens>>
Gaining "super" teller privilages
Accessing info on "super" teller procedures
<<include>>
<<include>>
<<include>>
<<include>>
Hacker
System admin.
System administration
<<threatens>>
<<threatens>>
Gaining access to the server for bank applications
<<include>>
Intruding the bank's network

Figure 3. MUC for the bank hack case used in the experiments (which is why no mitigations are shown).

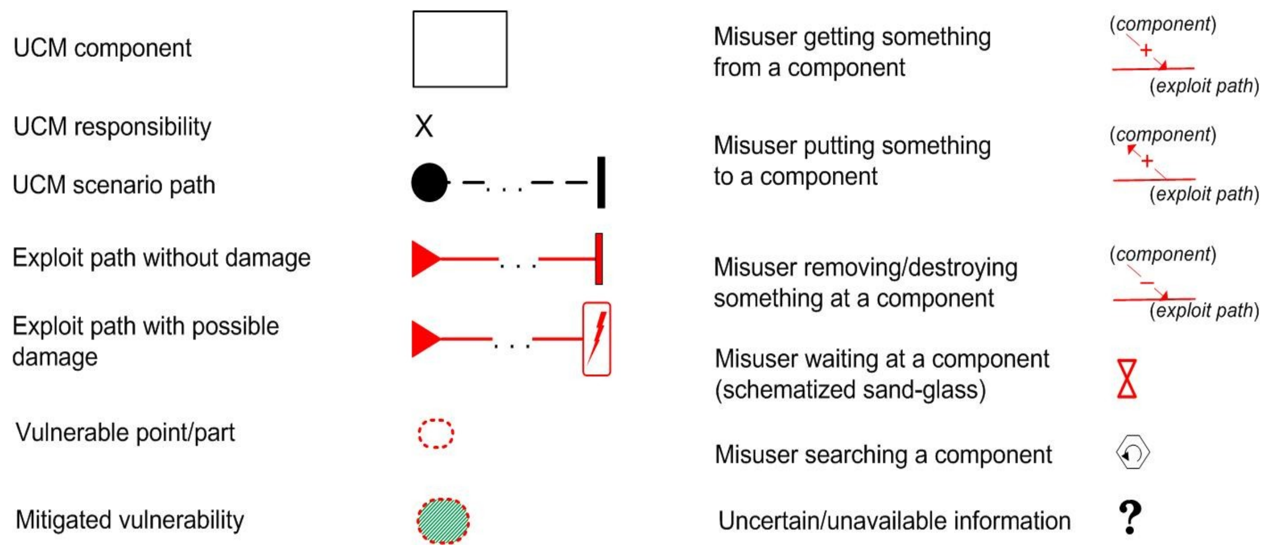

UCM component
UCM responsibility
X
UCM scenario path
Exploit path without damage
Exploit path with possible damage
Vulnerable point/part
Mitigated vulnerability
Misuser getting something from a component
(component)
+
(exploit path)
Misuser putting something to a component
(component)
+
(exploit path)
Misuser removing/destroying something at a component
(component)
–
(exploit path)
Misuser waiting at a component (schematized sand-glass)
Misuser searching a component
Uncertain/unavailable information
?

Figure 4. The MUCM notation, with regular UCM symbols in black and symbols that describe misuse in red.

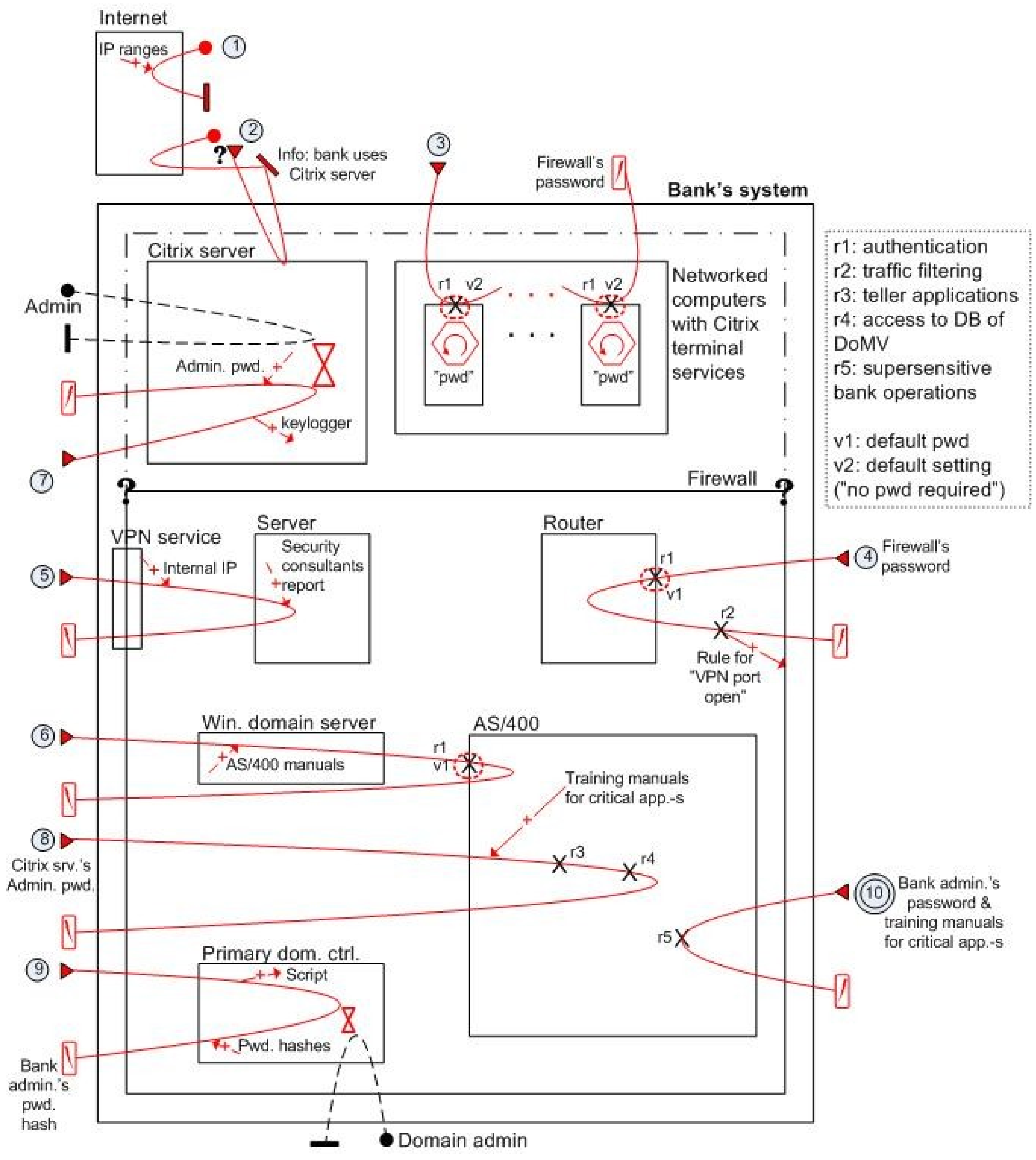


Figure 5. MUCM for the bank hack case used in the experiments.

Case order

H1
H2
H3
H4
H5
H6
H7
H8
H9
H1 1
H1 1

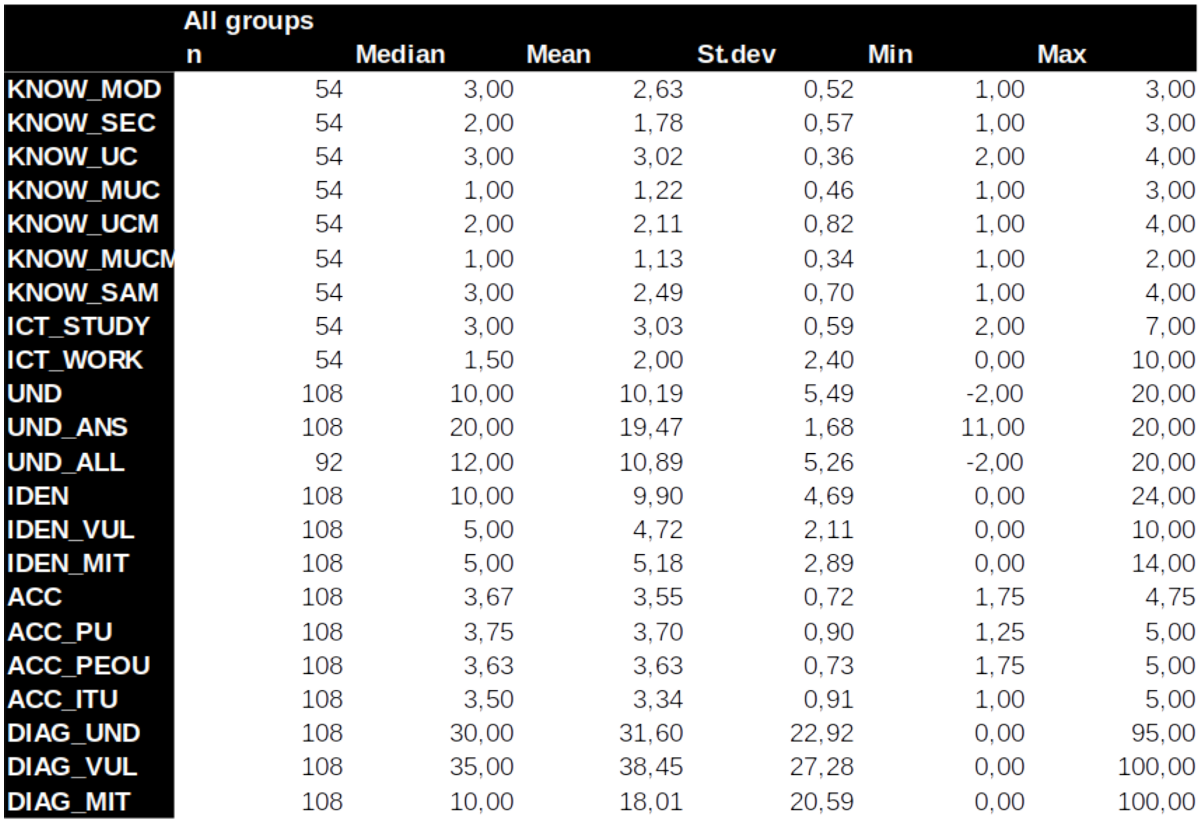
Name	Explanation
EXP	Which of the two experiments, either EXP1 or EXP2.
TECH	The *technique* used in that part of the experiment, either MUCM or MUC-SAD.
CASE	The *case* used in that part of the experiment, either BANK (the bank hack) or PEN (the penetration test).
PERIOD	Which of the two periods in EXP2, either PERIOD1 or PERIOD2.
KNOW_MOD, KNOW_THR, KNOW_UC, KNOW_MUC, KNOW_UCM, KNOW_MUCM, KNOW_SAD	The participants' self-assessed knowledge about *systems modelling* (KNOW_MOD), *threat analysis* (KNOW_THR), *use cases* (KNOW_UC), *misuse cases* (KNOW_MUC), *use case maps* (KNOW_UCM), *misuse case maps* (KNOW_MUCM), and *system architecture diagrams* (KNOW_SAD) on a 5-point scale, where 1 is "Never heard of it" and 5 is "Expert".
ICT_STUDY	The participants' self-reported *semesters* of ICT-studies.
ICT_JOB	The participants' self-reported *years* of ICT-relevant work experience.
UND, UND_ANS, UND_ALL	UND is the sum of scores for the all twenty statements about the problem cases, when a correct assessment is scored 1 and a wrong one -1, no assessment scored 0. UND_ANS is the number of assessments scored by a participant. UND_ALL is the sum of scores for only those participants who assessed all 20 statements.
IDEN, IDEN_VUL, IDEN_MIT	The sum of unique vulnerabilities and mitigations (IDEN) identified by the participants, and separate numbers of unique vulnerabilities (IDEN_VUL) and mitigations (IDEN_MIT).
ACC, ACC_PU, ACC_PEOU, ACC_ITU	Mean scores on the 5-point Likert scales for all the twelve statements about the how the techniques used were perceived (ACC) and for the groups of four statements about perceived usefulness (ACC_PU), perceived ease of use (ACC_PEOU), and intention to use (ACC_ITU) of the techniques.
DIAG_UND, DIAG_VUL, DIAG_MIT	The estimated extent of the use of diagram(s) when deciding about the statements in percentage, as compared to memory (DIAG_UND), when identifying vulnerabilities in percentage, as compared to memory (DIAG_VUL), and when identifying mitigations in percentage, as compared to memory and already identified vulnerabilities (DIAG_VUL).

Table 3: Variables used in the experiments

	Exp.	Alternative hypothesis	Null hyp.	Concl.
H1	**Both**	Participants using MUCM will understand an intrusion better than participants using MUC and SAD separately.	UND[MUCM] > UND[MUC-SAD]	***Accept***
H2	**Both**	Participants using MUCM will identify more security issues for an intrusion than participants using MUC and SAD separately.	IDEN[MUCM] > IDEN[MUC-SAD]	*Reject*
H3	**Both**	Participants using MUCM will perceive their technique differently from participants using MUC and SAD separately.	ACC[MUCM] = ACC[MUC-SAD]	***Accept***
H4	**Both**	Participants who understand an intrusion better using a technique will also identify more security issues for that intrusion using the technique.	UND[*tech*] ~ IDEN[*tech*]	***Accept***
H5	**Both**	Participants who understand an intrusion better using a technique will also perceive that technique more positively.	UND[*tech*] ~ ACC[*tech*]	***Accept***
H6	**Both**	Participants who identify more security issues for an intrusion using a technique will also perceive that technique more positively.	IDEN[*tech*] ~ ACC[*tech*]	***Accept***
H7	**EXP2**	Participants using MUCM will use diagrams more than participants using MUC and SAD separately.	UND[MUCM] > UND[MUC-SAD]	***Accept***
H8	**EXP2**	Participants who use diagrams more when using a technique will also better understand the intrusion using that technique.	DIAG[*tech*] ~ UND[*tech*]	***Accept***
H9	**EXP2**	Participants who use diagrams more when using a technique will also identify more security issues for the intrusion using that technique.	DIAG[*tech*] ~ IDEN[*tech*]	***Accept***
H10	**EXP2**	Participants who use diagrams more when using a technique will also perceive that technique more positively.	DIAG[*tech*] ~ ACC[*tech*]	***Accept***

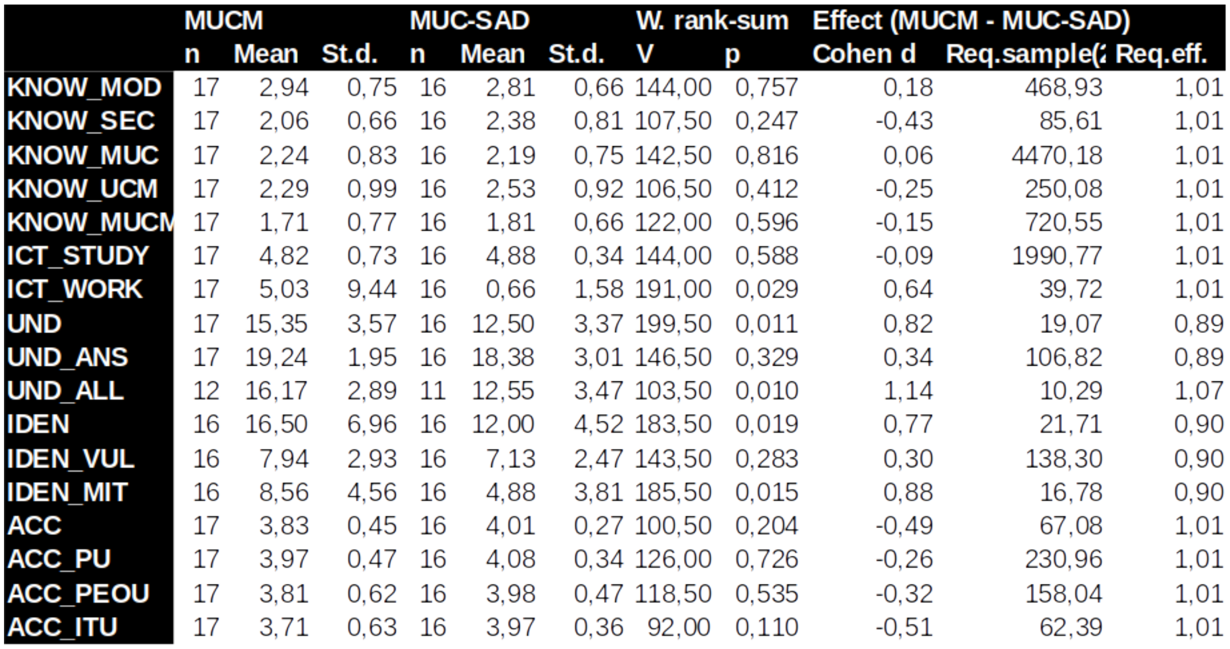
	Both groups					
	n	Median	Mean	St.dev	Min	Max
KNOW_MOD	33	3,00	2,88	0,70	1,00	4,00
KNOW_SEC	33	2,00	2,21	0,74	0,00	4,00
KNOW_MUC	33	2,00	2,21	0,78	1,00	3,00
KNOW_UCM	33	3,00	2,41	0,95	1,00	4,00
KNOW_MUCM	33	2,00	1,76	0,71	1,00	3,00
ICT_STUDY	33	5,00	4,85	0,57	2,00	5,00
ICT_WORK	33	0,00	2,91	7,12	0,00	36,00
UND	33	14,00	13,97	3,71	6,00	20,00
UND_ANS	33	20,00	18,82	2,52	10,00	20,00
UND_ALL	23	14,00	14,43	3,62	8,00	20,00
IDEN	32	13,00	14,25	6,21	3,00	33,00
IDEN_VUL	32	8,00	7,53	2,70	3,00	15,00
IDEN_MIT	32	7,50	6,72	4,54	0,00	18,00
ACC	33	3,92	3,92	0,38	2,92	4,67
ACC_PU	33	4,00	4,02	0,41	3,25	4,75
ACC_PEOU	33	4,00	3,89	0,55	2,50	4,75
ACC_ITU	33	3,75	3,83	0,53	2,50	5,00

Table *5*: Participant backgrounds in EXP1.



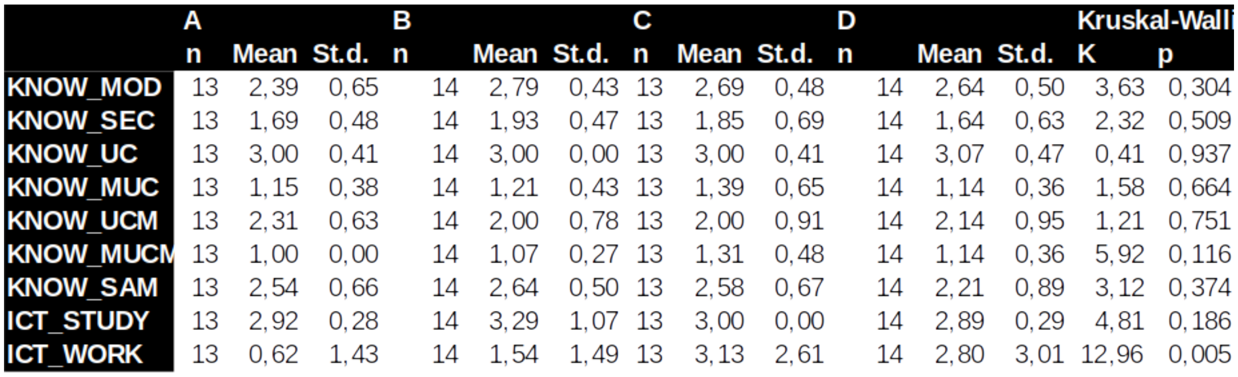
	All groups					
	n	Median	Mean	St.dev	Min	Max
KNOW_MOD	54	3,00	2,63	0,52	1,00	3,00
KNOW_SEC	54	2,00	1,78	0,57	1,00	3,00
KNOW_UC	54	3,00	3,02	0,36	2,00	4,00
KNOW_MUC	54	1,00	1,22	0,46	1,00	3,00
KNOW_UCM	54	2,00	2,11	0,82	1,00	4,00
KNOW_MUCM	54	1,00	1,13	0,34	1,00	2,00
KNOW_SAM	54	3,00	2,49	0,70	1,00	4,00
ICT_STUDY	54	3,00	3,03	0,59	2,00	7,00
ICT_WORK	54	1,50	2,00	2,40	0,00	10,00
UND	108	10,00	10,19	5,49	-2,00	20,00
UND_ANS	108	20,00	19,47	1,68	11,00	20,00
UND_ALL	92	12,00	10,89	5,26	-2,00	20,00
IDEN	108	10,00	9,90	4,69	0,00	24,00
IDEN_VUL	108	5,00	4,72	2,11	0,00	10,00
IDEN_MIT	108	5,00	5,18	2,89	0,00	14,00
ACC	108	3,67	3,55	0,72	1,75	4,75
ACC_PU	108	3,75	3,70	0,90	1,25	5,00
ACC_PEOU	108	3,63	3,63	0,73	1,75	5,00
ACC_ITU	108	3,50	3,34	0,91	1,00	5,00
DIAG_UND	108	30,00	31,60	22,92	0,00	95,00
DIAG_VUL	108	35,00	38,45	27,28	0,00	100,00
DIAG_MIT	108	10,00	18,01	20,59	0,00	100,00

Table *6*: Participant backgrounds in EXP2.



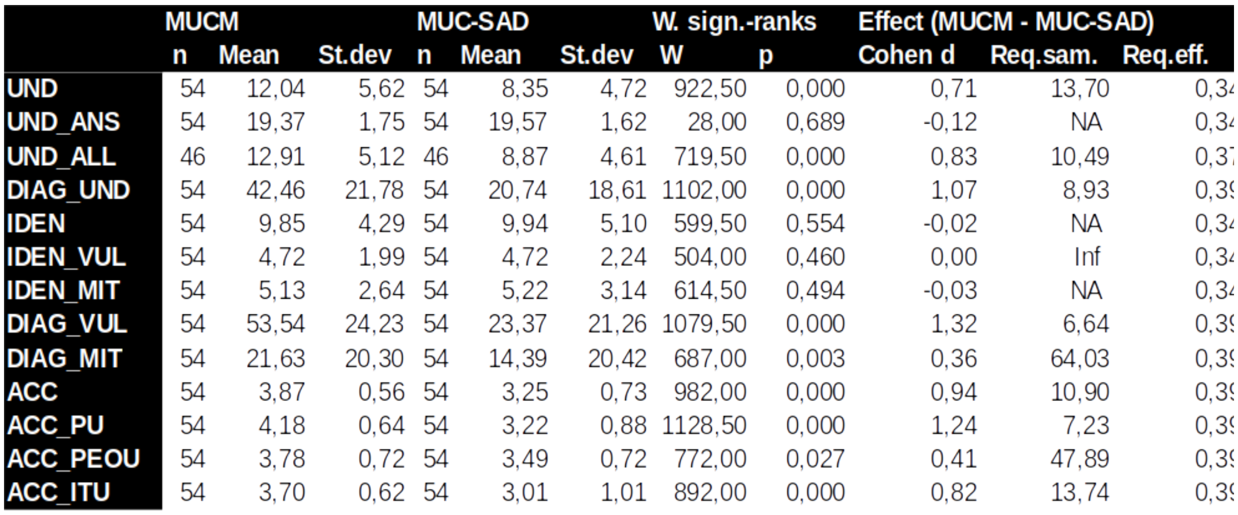
	MUCM			MUC-SAD			W. rank-sum		Effect (MUCM - MUC-SAD)		
	n	Mean	St.d.	n	Mean	St.d.	V	p	Cohen d	Req.sample(	Req.eff.
KNOW_MOD	17	2,94	0,75	16	2,81	0,66	144,00	0,757	0,18	468,93	1,01
KNOW_SEC	17	2,06	0,66	16	2,38	0,81	107,50	0,247	-0,43	85,61	1,01
KNOW_MUC	17	2,24	0,83	16	2,19	0,75	142,50	0,816	0,06	4470,18	1,01
KNOW_UCM	17	2,29	0,99	16	2,53	0,92	106,50	0,412	-0,25	250,08	1,01
KNOW_MUCM	17	1,71	0,77	16	1,81	0,66	122,00	0,596	-0,15	720,55	1,01
ICT_STUDY	17	4,82	0,73	16	4,88	0,34	144,00	0,588	-0,09	1990,77	1,01
ICT_WORK	17	5,03	9,44	16	0,66	1,58	191,00	0,029	0,64	39,72	1,01
UND	17	15,35	3,57	16	12,50	3,37	199,50	0,011	0,82	19,07	0,89
UND_ANS	17	19,24	1,95	16	18,38	3,01	146,50	0,329	0,34	106,82	0,89
UND_ALL	12	16,17	2,89	11	12,55	3,47	103,50	0,010	1,14	10,29	1,07
IDEN	16	16,50	6,96	16	12,00	4,52	183,50	0,019	0,77	21,71	0,90
IDEN_VUL	16	7,94	2,93	16	7,13	2,47	143,50	0,283	0,30	138,30	0,90
IDEN_MIT	16	8,56	4,56	16	4,88	3,81	185,50	0,015	0,88	16,78	0,90
ACC	17	3,83	0,45	16	4,01	0,27	100,50	0,204	-0,49	67,08	1,01
ACC_PU	17	3,97	0,47	16	4,08	0,34	126,00	0,726	-0,26	230,96	1,01
ACC_PEOU	17	3,81	0,62	16	3,98	0,47	118,50	0,535	-0,32	158,04	1,01
ACC_ITU	17	3,71	0,63	16	3,97	0,36	92,00	0,110	-0,51	62,39	1,01

Table *7*: Wilcoxon rank-sum test for difference between the MUCM and MUC-SAD groups in EXP1.



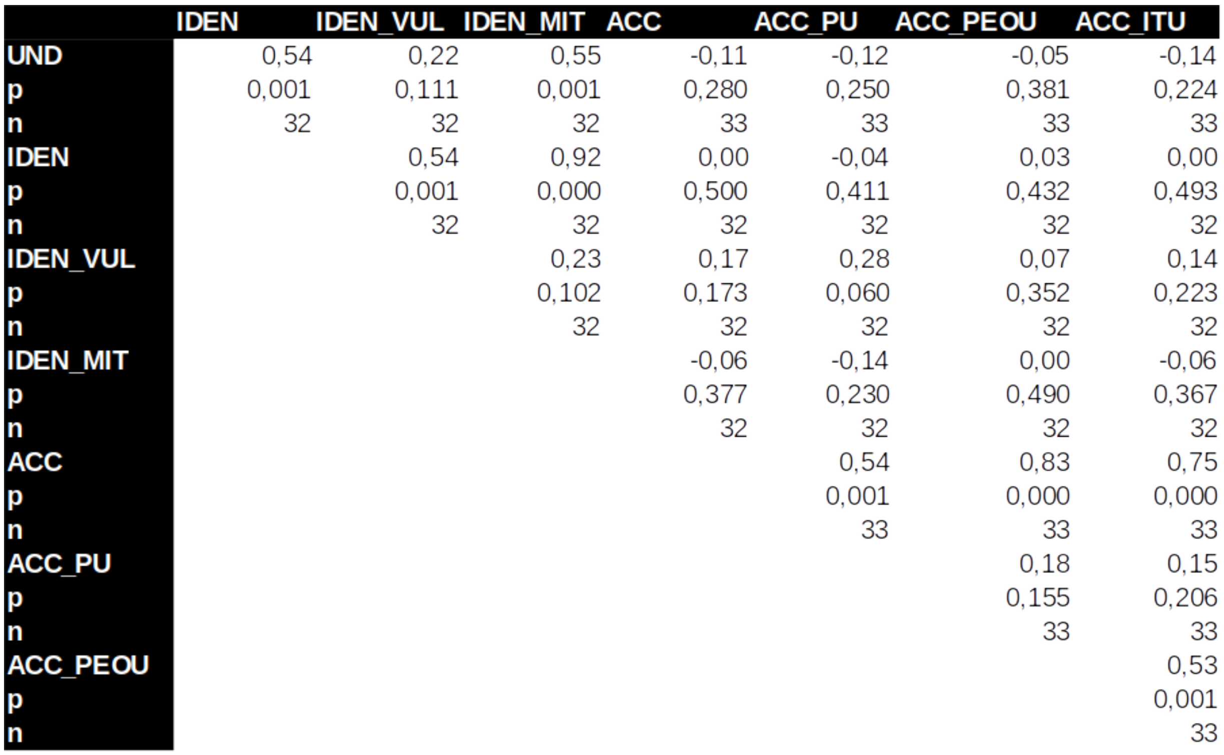
	A			B			C			D			Kruskal-Walli	
	n	Mean	St.d.	n	Mean	St.d.	n	Mean	St.d.	n	Mean	St.d.	K	p
KNOW_MOD	13	2,39	0,65	14	2,79	0,43	13	2,69	0,48	14	2,64	0,50	3,63	0,304
KNOW_SEC	13	1,69	0,48	14	1,93	0,47	13	1,85	0,69	14	1,64	0,63	2,32	0,509
KNOW_UC	13	3,00	0,41	14	3,00	0,00	13	3,00	0,41	14	3,07	0,47	0,41	0,937
KNOW_MUC	13	1,15	0,38	14	1,21	0,43	13	1,39	0,65	14	1,14	0,36	1,58	0,664
KNOW_UCM	13	2,31	0,63	14	2,00	0,78	13	2,00	0,91	14	2,14	0,95	1,21	0,751
KNOW_MUCM	13	1,00	0,00	14	1,07	0,27	13	1,31	0,48	14	1,14	0,36	5,92	0,116
KNOW_SAM	13	2,54	0,66	14	2,64	0,50	13	2,58	0,67	14	2,21	0,89	3,12	0,374
ICT_STUDY	13	2,92	0,28	14	3,29	1,07	13	3,00	0,00	14	2,89	0,29	4,81	0,186
ICT_WORK	13	0,62	1,43	14	1,54	1,49	13	3,13	2,61	14	2,80	3,01	12,96	0,005

Table *8*: Kruskal-Wallis test for difference between backgrounds of the four groups in EXP2.



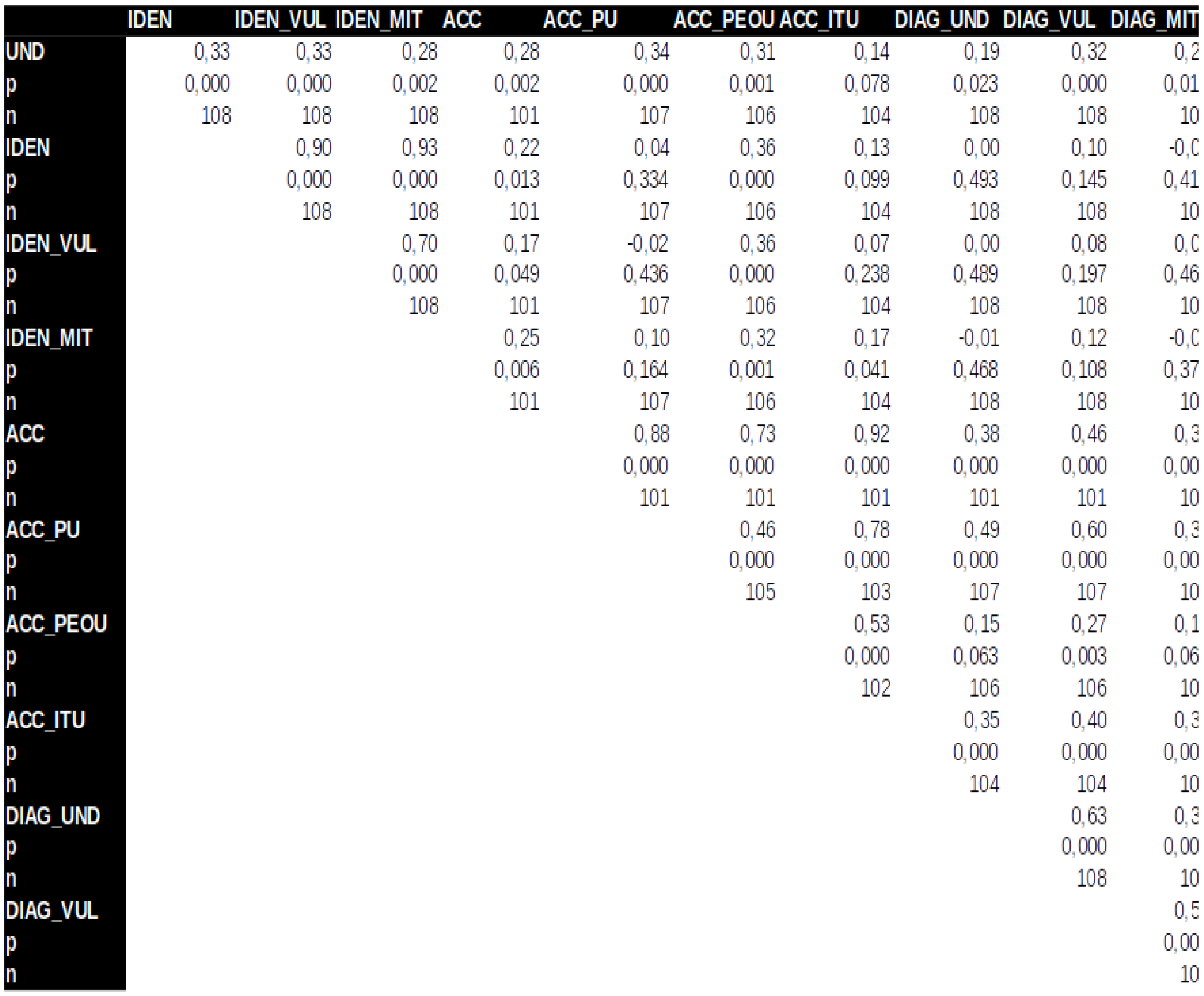
	MUCM			MUC-SAD			W. sign.-ranks		Effect (MUCM - MUC-SAD)		
	n	Mean	St.dev	n	Mean	St.dev	W	p	Cohen d	Req.sam.	Req.eff.
UND	54	12,04	5,62	54	8,35	4,72	922,50	0,000	0,71	13,70	0,34
UND_ANS	54	19,37	1,75	54	19,57	1,62	28,00	0,689	-0,12	NA	0,34
UND_ALL	46	12,91	5,12	46	8,87	4,61	719,50	0,000	0,83	10,49	0,37
DIAG_UND	54	42,46	21,78	54	20,74	18,61	1102,00	0,000	1,07	8,93	0,39
IDEN	54	9,85	4,29	54	9,94	5,10	599,50	0,554	-0,02	NA	0,34
IDEN_VUL	54	4,72	1,99	54	4,72	2,24	504,00	0,460	0,00	Inf	0,34
IDEN_MIT	54	5,13	2,64	54	5,22	3,14	614,50	0,494	-0,03	NA	0,34
DIAG_VUL	54	53,54	24,23	54	23,37	21,26	1079,50	0,000	1,32	6,64	0,39
DIAG_MIT	54	21,63	20,30	54	14,39	20,42	687,00	0,003	0,36	64,03	0,39
ACC	54	3,87	0,56	54	3,25	0,73	982,00	0,000	0,94	10,90	0,39
ACC_PU	54	4,18	0,64	54	3,22	0,88	1128,50	0,000	1,24	7,23	0,39
ACC_PEOU	54	3,78	0,72	54	3,49	0,72	772,00	0,027	0,41	47,89	0,39
ACC_ITU	54	3,70	0,62	54	3,01	1,01	892,00	0,000	0,82	13,74	0,39

Table *9*: Wilcoxon signed-ranks test for difference between the main variables when using MUCM and MUC-SAD in EXP2.



	IDEN	IDEN_VUL	IDEN_MIT	ACC	ACC_PU	ACC_PEOU	ACC_ITU
UND	0,54	0,22	0,55	-0,11	-0,12	-0,05	-0,14
p	0,001	0,111	0,001	0,280	0,250	0,381	0,224
n	32	32	32	33	33	33	33
IDEN		0,54	0,92	0,00	-0,04	0,03	0,00
p		0,001	0,000	0,500	0,411	0,432	0,493
n		32	32	32	32	32	32
IDEN_VUL			0,23	0,17	0,28	0,07	0,14
p			0,102	0,173	0,060	0,352	0,223
n			32	32	32	32	32
IDEN_MIT				-0,06	-0,14	0,00	-0,06
p				0,377	0,230	0,490	0,367
n				32	32	32	32
ACC					0,54	0,83	0,75
p					0,001	0,000	0,000
n					33	33	33
ACC_PU						0,18	0,15
p						0,155	0,206
n						33	33
ACC_PEOU							0,53
p							0,001
n							33

Table *10*: Spearman correlations between the main variables in EXP1, both groups pooled.



	IDEN	IDEN_VUL	IDEN_MIT	ACC	ACC_PU	ACC_PEOU	ACC_ITU	DIAG_UND	DIAG_VUL	DIAG_MIT
UND	0,33	0,33	0,28	0,28	0,34	0,31	0,14	0,19	0,32	0,2
p	0,000	0,000	0,002	0,002	0,000	0,001	0,078	0,023	0,000	0,01
n	108	108	108	101	107	106	104	108	108	10
IDEN		0,90	0,93	0,22	0,04	0,36	0,13	0,00	0,10	-0,0
p		0,000	0,000	0,013	0,334	0,000	0,099	0,493	0,145	0,41
n		108	108	101	107	106	104	108	108	10
IDEN_VUL			0,70	0,17	-0,02	0,36	0,07	0,00	0,08	0,0
p			0,000	0,049	0,436	0,000	0,238	0,489	0,197	0,46
n			108	101	107	106	104	108	108	10
IDEN_MIT				0,25	0,10	0,32	0,17	-0,01	0,12	-0,0
p				0,006	0,164	0,001	0,041	0,468	0,108	0,37
n				101	107	106	104	108	108	10
ACC					0,88	0,73	0,92	0,38	0,46	0,3
p					0,000	0,000	0,000	0,000	0,000	0,00
n					101	101	101	101	101	10
ACC_PU						0,46	0,78	0,49	0,60	0,3
p						0,000	0,000	0,000	0,000	0,00
n						105	103	107	107	10
ACC_PEOU							0,53	0,15	0,27	0,1
p							0,000	0,063	0,003	0,06
n							102	106	106	10
ACC_ITU								0,35	0,40	0,3
p								0,000	0,000	0,00
n								104	104	10
DIAG_UND									0,63	0,3
p									0,000	0,00
n									108	10
DIAG_VUL										0,5
p										0,00
n										10

Table *11*: Spearman correlations between the main variables in EXP2, all measures pooled.