predict
prioritise
prevent

TRESPASS

Technology-supported Risk Estimation by Predictive Assessment of Socio-technical Security

Deliverable D5.3.3

Best Practices for Model Maintenance

Project:	$TRE_S PASS$
Project Number:	ICT-318003
Deliverable:	D5.3.3
Title:	Best Practices for Model Maintenance
Version:	1.0
Confidentiality:	Public
Editor:	Olga Gadyatskaya
Cont. Authors:	M. Ford, O. Gadyatskaya, D. Gollmann, R.R. Hansen, D. Ionita, H. Jonkers, R. Kumar, A. Lenin, M. Martins, C. Muller, S. Muller, W. Pieters, C.W. Probst, K.J. Syauta, A. Tanner, J. Willemson
Date:	2016-10-31

SEVENTH FRAMEWORK PROGRAMME

Part of the Seventh Framework Programme
Funded by the EC-DG CONNECT

Members of the TRESPASS Consortium

1.	University of Twente	UT	The Netherlands
2.	Technical University of Denmark	DTU	Denmark
3.	Cybernetica	CYB	Estonia
4.	GMV Portugal	GMVP	Portugal
5.	GMV Spain	GMVS	Spain
6.	Royal Holloway University of London	RHUL	United Kingdom
7.	itrust consulting	ITR	Luxembourg
8.	Goethe University Frankfurt	GUF	Germany
9.	IBM Research	IBM	Switzerland
10.	Delft University of Technology	TUD	The Netherlands
11.	Hamburg University of Technology	TUHH	Germany
12.	University of Luxembourg	UL	Luxembourg
13.	Aalborg University	AAU	Denmark
14.	Consult Hyperion	CHYP	United Kingdom
15.	BizzDesign	BD	The Netherlands
16.	Deloitte	DELO	The Netherlands
17.	Lust	LUST	The Netherlands

Document History

Authors		
Partner	Name	Chapters
UL	Olga Gadyatskaya	All
CYB	Alexandr Lenin	6
	Jan Willemson	2
UT	Rajesh Kumar, Dan Ionita	6
DTU	Christian W. Probst	6
ITR	Miguel Martins, Cédric Muller, Steve Muller	7
TUD	Wolter Pieters, Kevin Joseph Syauta	3
TUHH	Dieter Gollmann	2
AAU	René Rydhof Hansen	6
CHYP	Margaret Ford	1
BD	Henk Jonkers	6
IBM	Axel Tanner	5

Quality assurance		
Role	Name	Date
Editor	Olga Gadyatskaya (UL)	2016-10-31
Reviewer	Bob van Kan (DELO)	2016-10-10
Reviewer	Rajesh Kumar (UT)	2016-10-14
Task leader	Jan Willemson (CYB)	2016-10-31
WP leader	Jan Willemson (CYB)	2016-10-31
Coordinator	Pieter Hartel (UT)	2016-10-31

Circulation	
Recipient	Date of submission
Project Partners	2016-09-30
European Commission	2016-10-31

Acknowledgement: *The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 318003 (TRESPASS). This publication reflects only the authors' views and the Union is not liable for any use that may be made of the information contained herein.*

Contents

List of Figures

Management Summary

Key takeaways This document is the public deliverable D5.3.3 *Best Practices for Model Maintenance*. The key takeaways from this document are:

- Best practices for the TRE$_S$PASS models maintenance were developed with two goals in mind: *continuous maintenance of complex models* and *facilitation of model maintenance in the presence of change*. These two goals have been identified by studying the state-of-the-art in model-driven engineering and lessons learned from relevant large EU projects, and were validated by the practical needs arising from TRE$_S$PASS process implementations (i.e., in the project's case studies).
- TRE$_S$PASS leverages state-of-the-art techniques in model-driven engineering such as model transformation and traceability links maintenance to ensure automatic co-evolution of the critical models in TRE$_S$PASS socio-technical risk assessment. Specifically, updates are often propagated automatically across the TRE$_S$PASS models and the analysis process.
- New security situations have a strong impact on organizations and their security posture. Therefore, handling new security situations and maintaining models in the presence of change that these new situations bring is an important requirement for the TRE$_S$PASS process. We report on our approaches for handling dynamic scenarios, facilitating model updates, and our take on dynamic risk analysis.
- The TRE$_S$PASS process now incorporates a well-defined approach to model maintenance, as the project has developed the means to maintain analyzability of all the models involved. In line with the TRE$_S$PASS tool ecosystem that has developed around many standalone tools and methodologies and aggregates them according to the analyst's needs, we have developed techniques for maintaining the underlying models that do not depend on each other, and can be efficiently combined.

1 Introduction

1.1 Goals

The TRE$_S$PASS process involves multiple modelling languages and approaches. All its tasks, from eliciting business scenarios with LEGO or formalizing the socio-technical landscape of a company to analyzing the resultant socio-technical scenarios in detail and sharing results of analysis across organisations, involve models. Thus, we have to understand how to create, store, share and maintain complex models.

Task T5.3 *Model creation, sharing and maintenance* focuses on the best practices for handling models in the TRE$_S$PASS process. Previous project deliverables D5.3.1 *Abstraction levels for model sharing* (The TRE$_S$PASS Project, D5.3.1, 2013), and D5.3.2 *Best practices for model creation and sharing* (The TRE$_S$PASS Project, D5.3.2, 2015) in this task have mostly focused on studying the best practices for model creation and sharing and identifying the methodologies for model creation and sharing in TRE$_S$PASS. As the project currently has reached its maturity and has developed a rich collection of modelling languages and tools, we also set out to systematically study how to maintain our models and how to ensure the analysis results' validity in presence of new situations. The main goal of this document is to present our findings and define strategies for model maintenance in TRE$_S$PASS. The techniques we report are *best practices* in the sense that they 1) leverage state-of-art techniques in model-driven engineering; 2) were validated by the project in the numerous validation exercises done within the case studies and/or by involving suitable domain experts.

We focus on two important aspects of model maintenance:

- *Continuous maintenance of complex models* is required for the TRE$_S$PASS modelling artefacts in order to enable efficient knowledge building and sharing.
- *Facilitation of model maintenance in the presence of change* is needed to ensure that if a new risk emerges, the analyst will not need to re-start the whole process from scratch.

Notice that these two goals are not orthogonal, but they are complementary and can often be ensured by applying the same techniques. We now explain the motivations behind these two aspects in more detail.

1.2 Maintenance and sharing of complex models

TRESPASS relies on a model-based process, as it incorporates socio-technical and attack models at different granularities and introduces transformations from socio-technical models into attack models (The TRESPASS Project, D5.4.2, 2016). The end-users of the TRESPASS process and the toolset will not use the project's results only once, but will continuously improve them and will apply them in diverse contexts. Thus, it is evident that we need to provide the means to maintain TRESPASS models, share them across applications, and ensure the overall usability of our modeling infrastructure.

1.2.1 Continuous knowledge building

Nowadays the security community clearly recognises that systematization of knowledge is required to support professionals in day-to-day tasks. This systematization of knowledge manifests itself in the security domain through proliferation of various databases (e.g., the CVE system for common vulnerabilities and exposures (MITRE, 2016b)), catalogues (such as CAPEC (MITRE, 2016a)), and standards (NIST SP 800-53, ISO 27xxx, etc.). These systems support accumulation of the knowledge and expertise of different people, enable efficient sharing of this knowledge, and allow newcomers in the field to get quickly familiarized with the problem domain.

In TRESPASS we have applied the knowledge building approach by introducing the various libraries (e.g., the attack pattern library and the attacker profile library (The TRESPASS Project, D5.4.2, 2016)) and the TRESPASS information system (The TRESPASS Project, D2.4.1, 2016) (also referred to as the knowledge base in this deliverable) that can be continuously improved by the analysts. Furthermore, the socio-technical model proposed in TRESPASS is also modular, and the analyst can create and reuse patterns encapsulating complex systems and behaviours (e.g., a firewall sub-model), thus enabling continuous simplification for the modelling stages in the subsequent risk assessment exercises (The TRESPASS Project, D1.3.4, 2016). Furthermore, model instance versioning can often be required for reviews and audit. We have developed the information system to support model instance versioning relying on the git approach.

1.2.2 Model sharing

Another obvious benefit of the knowledge-building paradigm we just described is that it facilitates knowledge sharing. Indeed, once a comprehensive database of threats has been established in a particular domain, all stakeholders from this domain can benefit from this database. While the modular approach chosen by TRESPASS enables efficient sharing of information, we have further enhanced it by applying the model transformation paradigm. For example, TRESPASS has proposed a meta-model for attack trees that enables efficient transformations for attack tree models expressed using different notations (The TRESPASS Project, D3.5.1, 2016). This means, for instance, that an attack tree created by the Treemaker (The TRESPASS Project, D3.4.1, 2014) can be processed and

analyzed by the various independent project tools (e.g., the ADTool or the ATTop) (The TRESPASS Project, D3.4.2, 2016), though these tools rely on different dialects of attack trees.

1.2.3 Maintenance support

Last but not least, model maintenance techniques need to be usable. When dealing with complex models (e.g. a socio-technical model of a large organization), if the analyst could not see easily how it relates to other TRESPASS process elements (e.g., attack scenarios), then it would be very difficult for her to reason about various options and perform "what-if" analysis. Thus, we have performed a large study about application of traceability techniques for TRESPASS artifacts. We report this study in the present deliverable.

1.3 New security situations as a result of change

New security situations emerge everyday. These new situations can be triggered by changes in the organisation itself (internal changes), or by changes in the environment (external), for example, changes in the threat landscape and changes in legal and regulatory frameworks. Thus, TRESPASS has developed several techniques for an efficient treatment of changes in the organisation and its environment.

1.3.1 Changes in the organisation

Each organisation is not static, but it constantly evolves. As the TRESPASS process is based on modelling organisations, we need to accommodate organisational changes in the process and to support updates to organisational models and the analysis results. Relevant organisational changes take many forms. Therefore, we need to consider the full spectrum of evolution: from updates to the organisational infrastructure (including physical and digital components of it) to the human resources and organisational hierarchy changes.

For example, companies are often driven to switch to the latest technologies in order to save money and optimize resources. At the same time, new advances in technology, such as Cloud Computing, Internet of Things, or Mobile Computing, lead to new vulnerabilities and cyber-security issues (Georgia Institute of Technology, 2014). Therefore, for us it is essential to be able to model these new technologies and to analyze them in the TRESPASS tool suite. To cope with this requirement, we have designed the TRESPASS socio-technical model to be *extensible*, i.e., new features and elements can be added to the socio-tehnical modelling language (The TRESPASS Project, D1.3.2, 2015).

At the same time, some changes in organisations that are more regular, e.g., human resources fluctuations or infrastructure changes, also need to be reflected in the model of

the organisation. To support model elements that change frequently we have considered dynamic features in socio-technical models (The TRESPASS Project, D1.3.3, 2015).

1.3.2 Changes in the environment

Another important category of relevant changes is changes in the environment, or external changes. This type of change imposes constraints on the organisation that need to be accounted for in the security risk analysis process.

Changes in the threat landscape Every day brings new security concerns for organisations. We have already mentioned new technologies as a source of security problems, but the *threat landscape* is a more broad term that captures existing threats to an organisation and how these threats evolve (ENISA, 2012). Therefore, evolution in the threat landscape requires constant attention from the organisation. Indeed, in the risk assessment process the organisation has identified some important assets, relevant vulnerabilities, possible threat agents, potential attack avenues and the expected likelihood of attacks. If one of these identified values changes, the risk analysis results may also change drastically.

Considering the recent Symantec Internet Cyber Threat report (Symantec, 2015) as an example of an overview of changes in the general threat landscape, we can see that many of the most prominent threats have a "social flavor" to them, for example, the scams executed via social media, or the targeted attacks. Therefore, the ability of the holistic TRESPASS process, which aggregates physical, digital and social domains, to react to the evolution of the threat landscape is indeed an important achievement.

Changes in legal and regulatory frameworks New legal and regulatory requirements often impose constraints on the security processes in the organisation. For example, the upcoming reform in the European Data Privacy Framework will require that companies notify their users individually in case of a data breach. It will also enable citizens to sue organisations in case of mishandling of their data (European Commission, 2015). Thus, organisations will need to improve their data protection schemes and arrange new channels to communicate with the end-users. Therefore, while compliance is not the core goal of TRESPASS, the TRESPASS process needs to be agile enough to be able to capture these updates to organisational security emerging as a result of the changes in legal requirements, and to accommodate them in the analysis process.

The main tools developed by TRESPASS for supporting model maintenance for changes in the environment are the flexibility of the socio-technical modelling language (as described above) and the ability to handle new technical and social data when the need arises (The TRESPASS Project, D2.4.1, 2016).

1.4 Foreground and background

TRE$_S$PASS has learned a great deal from the state of the art on model maintenance and results of relevant research projects. These results are background of the project. The majority of contributions in this deliverable are foreground of the project. Whenever appropriate, we give references to relevant TRE$_S$PASS deliverables that provide more details about the techniques presented. Those deliverables provide indications on the applicable foreground/background composition. We estimate that 90% of this deliverable are foreground TRE$_S$PASS results.

1.5 Document structure

The remainder of this deliverable is structured as follows. Chapter 2 gives an overview of lessons learned from relevant European projects. Chapter 3 presents our findings regarding application of traceability links maintenance, and Chapter 4 outlines TRE$_S$PASS results in model transformation. Chapter 5 summarises tasks and activities in TRE$_S$PASS that have to do with changing situations and approaches to handle such situations. Then, in Chapter 6, we provide a summary of the important models in the TRE$_S$PASS process and present the established practices for maintaining analysability of these models. In Chapter 7 we overview dynamic risk assessment as the means to account for dynamic changes in the environment as a part of the TRE$_S$PASS process. We summarize our main findings and techniques in Chapter 8. Finally, we conclude with Chapter 9.

2 Lessons Learned from EU Projects

In this chapter we list important findings from relevant European research projects. These lessons learned helped us to understand many requirements for model maintenance in complex security processes, such as the TRE$_S$PASS process, and they guided us towards establishing the best practices for model maintenance reported in this document.

2.1 Lessons Learned from SecureChange

SecureChange was a European research project funded by the EU under the FP7-ICT programme[1]. The main goal of the project was to propose mechanisms to holistically ensure security for the full software development lifecycle in the presence of change. Indeed, software is often affected by changes that can be introduced at each step of its existence: from requirements elicitation and architecture design to testing, verification, deployment and maintenance. The vast majority of the existing security-relevant mechanisms for each of these lifecycle steps do not take into account this changing nature of software artifacts. SecureChange has worked at each software lifecycle step to identify the nature of change that can be introduced there, and how to improve the existing techniques and methodologies to accommodate the change and to maintain security of the system throughout the change process. The methodologies considered by the project were model-based, and therefore, very relevant to the TRE$_S$PASS model-based process. In this section we set out to identify the lessons learned from the results of SecureChange (this was also one of the suggestions given by the Advisory Board members in 2014).

Models are the crucial part of the software development lifecycle, and they can be considered as the key artifacts that are produced, exchanged and maintained throughout the software system development process (Felderer et al., 2014). Indeed, each software engineering step deals with their own models, for example, requirements models, architecture models, risk models, and testing models. Models from one step can be consumed as inputs on the subsequent steps. In the same time, as reported in the article (Felderer et al., 2014) issued by the project participants, while these modelling artifacts are tightly coupled, the majority of methodologies work at a specific level and with one modelling language in isolation. The authors suggest that a more holistic approach, advocated by SecureChange, can be a solution to ensuring security throughout the full software development lifecycle. In this holistic approach, each modelling language should be equipped with traceability links to modelling languages at other layers. The traceability links should ensure consistent transformation of change information and assist in keeping

[1] http://www.securechange.eu/

the system secure under evolution. For more details on the SecureChange project's approach and results we refer the interested readers to the final publishable summary of the project (Massacci, 2012), where precise pointers to the relevant deliverables are given. All public project's deliverables are located at http://www.securechange.eu/content/deliverables.

A concrete example of a mutual evolution management approach established by SecureChange between two software engineering steps while maintaining the separation of modelling scopes is documented in (Paci, Massacci, Bouquet, & Debricon, 2012). The authors introduce an approach to orchestrate mutual evolution of requirements models and test models via a minimal shared interface. The interface defines a set of shared concepts that influence models in both requirements and testing domains. The shared concepts are linked with the modelling languages used in the domains. This set of concepts is minimal in the sense that the included concepts are selected as to respect the separation of concerns between the requirements and testing domain. These concepts can be *shared* or *mappable*. Shared concepts are those that have the same semantics in both the domains considered. Mappable concepts are concepts from one domain that can be mapped to concepts in another domain. For example, *requirement* is a shared concept, while the *goal* and *process* concepts from the requirements domain can be mapped into the test model. The orchestration across two domains follows a protocol that starts when a stakeholder requests a change in the system.

We can immediately see that the findings of SecureChange are relevant to the TRE$_S$PASS process, which is also model-based. In the TRE$_S$PASS process we start with a socio-technical system model and enrich it with data. The analysis process consumes the socio-technical model and other relevant models (e.g., attacker models from the Attacker Profile Library) and outputs potential attack scenarios formalised and visualised as attack tree models. Then the analysis also proposes countermeasures and mitigation strategies, which are reflected in the attack tree models and are propagated back to the socio-technical model. Afterwards the analysis can be re-launched to find potential attacks on the updated infrastructure. Complete version of the TRE$_S$PASS process is available in (The TRE$_S$PASS Project, D5.4.2, 2016). Yet, it is clear that there should be a coherent process to maintain traceability between the socio-technical model and the attack model that will allow to go back from the components identified as vulnerable in the analysis to the right place in the socio-technical model (The TRE$_S$PASS Project, D3.4.2, 2016). One of available the mechanisms for traceability in TRE$_S$PASS is the model elements identifiers. We report on our own traceability study for the TRE$_S$PASS process in Chapter 3.

2.2 Lessons Learned from SHIELDS

SHIELDS was a EU FP7 supported project running from 2008-01-01 to 2010-06-30[2]. It was similar to TRE$_S$PASS in some of the respects, even though it did not have the con-

[2] http://cordis.europa.eu/project/rcn/85431_en.html

cepts of system model, navigator map, etc. Rather, it was trying to perform the modelling and analysis on top the attack trees directly.

One of the innovations introduced by SHIELDS was to establish a library of trees corresponding to standard elementary attacks. A web-based platform to support tree distribution was developed and initially populated.

However, the SHIELDS project failed to attract a larger user community who would start using and contributing to this library. We conjecture that the main reason behind this failure was the lack of user-friendly model development environment and insufficient integration into real risk assessment processes. The SeaMonster software[3] was only capable of creating attack trees and misuse case diagrams, but this proved to be insufficient to attract real risk analysts, even with the ready-made elementary tree library.

As a result, the project is now in a dead state and even its webpage maintenance has been discontinued.

The main lesson to be learnt from the SHIELDS project is that taking an analysis-method-centric approach to risk assessment is not sufficient. It is hard to build a community around a toolset requiring the end user to work with the attack trees directly. While there definitely exist analysts for whom such an abstraction level is natural, their number is considerably more limited compared to those who'd like to think on the system level. This is why TRE$_S$-PASS has chosen a different viewpoint and started building the tools from the system modelling perspective, while the attack tree analysis happens mainly under the hood. At the same time, TRE$_S$PASS has acknowledged the benefits of a tree library that allows to build and share knowledge. We have applied this principle in the attack pattern library (The TRE$_S$PASS Project, D5.4.2, 2016).

2.3 Lessons Learned from ContainIT

ContainIT was a collaborative German research project (BMBF FKZ 13N11013) conducting a feasibility study for continuous risk assessment in the logistics sector with a specific focus on container transport (Sauff & Gollmann, 2012). Characteristic features that need to be considered in this domain are:

- The parties handling a container during its journey from source to eventual destination may be defined dynamically during transport.
- Besides the container that is moved physically, there are various required business documents, e.g. the bill of lading, that are also exchanged between the parties involved and serve as a basis for consistency checks.

The goal of the project was the design of a risk analysis scheme that would compute a risk rating from features such as the route taken, including deviations from the planned route or stay-overs at known trouble spots, incoherence in the documentation for a transport,

[3] http://sourceforge.net/projects/seamonster/

or the parties that had been handling the container. Risk analysis would have to cover physical attacks and attacks in cyberspace.

The main lesson learned was that a modelling methodology supporting this approach has to capture the physical flow of goods and the information flows between the parties involved in a given transport, and the interactions between the physical and information flows. The method adopted was based on swim-lane diagrams with time on the horizontal axis and parties on the vertical axis. Interactions between parties correspond to events crossing lane boundaries. Such swim-lane diagrams are convenient for mapping the results of interviews with domain experts and for identifying potential vulnerabilities by visual inspection.

The project considered straightforward computation of the overall dynamic risks from the likelihood estimations for applicable attacks given by experts. Considering the distributed and incomplete nature of the information about containers in the system, this approach was found to be quite successful (Sauff & Gollmann, 2012). Thus, one important consideration for TRESPASS is that very dynamic and distributed environments might require more lightweight, flexible risk assessment approaches. We report on our study of dynamic risk assessment applicability for TRESPASS in Chapter 7.1. Another important lesson learned from this project concerns expert opinions. The ContainIT project found that in rare cases widely divergent expert opinions were received, it could be usually explained by differences in the interpretations of a question by the experts. TRESPASS has investigated these issues for data gathering activities ((The TRESPASS Project, D2.2.3, 2016, 2016)).

3 Traceability Links Maintenance

TRESPASS has invested in furthering the state of the art in traceability in risk assessment, by studying how traceability could be incorporated in the TRESPASS tools and processes as an example of traceability in cyber risk management. This has resulted in the master's thesis (Syauta, 2016), with input from many consortium members through interviews. The text below is a selection and adaptation from (Syauta, 2016).

3.1 Traceability in different domains

Traceability as a notion is interpreted differently in various domains. In (Ammar, Benaissa, & Chabchoub, 2015), these different definitions are outlined. For example, in computing, traceability has to do with products and development process (IEEE, 1991). In quality management, traceability is more related with recorded identification of an article or an activity (International Organization for Standardization, 1994). In the domain of food industry, there are four different contexts in which the term traceability can be used (Moe, 1998):

1. Product: including its materials, its origin, its processing history, distribution and location after delivery
2. Data: including data calculations and generation generated data throughout the quality loop
3. Calibration: measuring equipment to standards, constants or properties, or reference materials
4. IT and programming: design and implementation back to system requirements.

Although these contexts are derived from food industry, the general idea of this typology can be applied to other sectors. The four items are not exclusive to food industry, hence lending themselves for use in other domains. The second and the fourth context (Data and IT and programming) will be discussed further in Section 3.1, as both of these contexts can be related to TRESPASS. The second context will be elucidated by taking a look at how traceability is defined in food industry, while the fourth context shall be elaborated by taking the example of software development domain.

Most of the works on traceability is dedicated to the domain of supply chain, especially food industry. In supply chain, traceability concerns the activities of capturing, recording, and transmitting information at each stage of the supply chain (Ammar et al., 2015). In

the food industry, traceability is viewed as a tool to “trace and follow" and retrieve information, as well as a record-keeping system by means of recorded identification (Aung & Chang, 2014). As an example, a framework for food traceability consisting of four elements (*product identification*, *data to trace*, *product routing* and *traceability tools*) is proposed in (Regattieri, Gamberi, & Manzini, 2007). Moe explains that there are two types of food traceability: *internal traceability* (which tracks internally in one of the steps in the chain) and *chain traceability* (which tracks a product batch and its history through a production chain) (Moe, 1998). Aung and Chang classify traceability on the basis of the direction in which the information is recalled in the chain: *backward traceability* or *tracing* (finding the *origin and characteristics* of a product from one or more criteria) and *forward traceability* or *tracking* (finding the *locality* of products from one or more criteria). Data traceability in food industry is mostly needed due to ethical and legal compliance (Moe, 1998), which includes official control of foodstuffs and liability for defect products (van Dorp, 2002). It is made possible through *item coding* and *information architecture* (van Dorp, 2002).

Traceability in software development has gone from the ability to trace requirements down to code (*requirements traceability*) (Seibel, 2011) to “the capability for tracing artefacts along a set of chained operations..." (*software traceability*) (Paige, Olsen, Kolovos, Zschaler, & Power, 2008). According to IEEE Guide to Software Requirements Specification, “A software requirements specification is traceable if the origin of each requirement is clear and if it facilitates the referencing of each requirement in future development or enhancement documentation." (“IEEE Guide for Software Requirements Specifications”, 1984). Traceability in model-driven engineering, for instance, can be established through *trace links* that connect different sources to their respective target elements (Olsen & Oldevik, 2007). Requirements traceability encompasses both *forward traceability* or the ability to trace components of a requirements specification to components of a design or of an implementation and *backward traceability*, or the ability to trace a requirement to its source that demands the requirements to be present (Wieringa, 1995). An example of integration of traceability into software development is the MoVE (Model Versioning and Evolution) project (The MoVE Project, 2013).

Definitions of traceability in food supply chain and in software development do display commonality to a certain extent. For example, both domains acknowledge the presence of both backward-looking and forward-looking traceability and that these two altogether are instrumental in defining an overall framework of traceability. Both domains also view that recorded identification or documentation is indispensable in orchestrating traceability in a system. Their difference lies in that food industry views traceability in the context of product and data, while software engineering views traceability in the context of requirements, according to the definitions given in (Moe, 1998).

As a final remark, traceability in cyberspace was explored extensively in a dissertation written by security researcher Richard Clayton. He defines traceability as the ability to trace events in cyberspace back to the ‘doer’ in the physical world (Clayton, 2005). In this sense, traceability has a close relation with attribution, and anonymity is understood as the absence of traceability. His work pivots on tracing back large-scale cyber crime activities, and not risk assessment.

3.2 Traceability in (cyber) risk assessment

Current state of the art on traceability in risk assessment connects traceability with the concept of *change*. Traceability can facilitate maintenance of risk models and can help preserve its validity as the target of analysis undergoes changes and evolves (Solhaug & Seehusen, 2014). In CORAS, this is also known as *evolutionary risk analysis* (Felici, Meduri, Solhaug, & Tedeschi, 2011; Lund, Solhaug, & Stølen, 2010). According to Lund et al, there are three different perspectives with regards to changes in a system (Lund, Solhaug, & Stølen, 2011). These are:

1. *Maintenance perspective*: Changes that accumulate over time.
2. *Before-after perspective*: Radical, but planned changes.
3. *Continuous evolution perspective*: Predictable changes as a function of time (gradual and steady evolution)[1]

In the maintenance perspective, both old risks (before changes accumulate) and current risks are taken into account. In the before-after perspective, current risks, future risks, and risks that occur due to change process are considered. Finally, the continuous evolution perspectives pays attention to evolving risks at specific points in time until the evolution fully takes place. The CORAS approach concentrates solely on the before-after perspective (Refsdal, Solhaug, & Stølen, 2015; Seehusen & Solhaug, 2012).

In the TRE$_S$PASS project, the concept of change is captured in (The TRE$_S$PASS Project, D1.3.3, 2015). The document assesses the dynamic features of the TRE$_S$PASS model, which are related to circumstantial or contextual changes that are of temporal nature. This is important because temporal changes might cause the chance of success of an attack to vary, which makes it important for the TRE$_S$PASS model to cope with such changes. The document also lists some dynamic features in the case studies. However, the deliverable is limited to only developing support for dynamic features, but not on tracking these dynamics in a system. Our suggestion is that TRE$_S$PASS model should focus on the *maintenance* perspective for the following reasons:

- The maintenance perspective assumes that previous risk analysis is documented and relevant changes are considered as input to derive the current risk picture (Lund et al., 2011). This is aligned with the TRE$_S$PASS case study on the cloud model, in which a preliminary risk picture is already available. The many data input tools (e.g. vulnerability databases, libraries) are beneficial in conducting an up-to-date risk assessment using existing data.
- The before-after perspective takes into account the risks due to the change process. This is beyond our scope, and is sometimes impossible to quantify. For example, the change process following the discovery of a new vulnerability is fuzzy. During that period the risk of being attacked by a zero-day exploit is present. However,

[1] Similar to the Plan-Do-Check-Act cycle in an ISMS (information security management system) governed by standards such as ISO 27001.

by definition a zero-day exploit is unknown to the vulnerable party, hence it is not possible to quantify its risks.

- The continuous evolution perspective is hard to envisage in our case since we cannot really predict changes that happen at a specific point of time based on solid forecasts or planned development. One of the main characteristics of cloud computing is rapid elasticity and continuous evolution is not associated with elasticity. Predicting how risks would evolve in such an environment is not plausible.

There exists no specific definition for traceability for risk assessment. A conceptual model for traceability in safety systems is instantiated in (Katta & Stalhane, 2010), but it hardly mentions aspects of risk. The closest approach to embed traceability in risk assessment is related to software development, especially to one specific methodology called *model-driven engineering*. For example, there have been various attempts to infuse traceability to the CORAS approach (which is fashioned after model-driven engineering) by building a trace model (Seehusen & Solhaug, 2012; Solhaug & Seehusen, 2014; Refsdal, Solhaug, & Stølen, 2015). In CORAS approach, a trace model is a set of pairs that include elements of the risk model and the model that contains elements of the system of interest (target model) which was cultivated by adopting an critical infrastructure system as an example. Although traceability is extensively discussed in CORAS, a formal definition for traceability is not specified.

Another attempt to integrate traceability in risk assessment is done with the Rinforzando tool, in which links between risk models and system engineering models can be maintained (Paul & Delande, 2011). A comparison between CORAS and Rinforzando is presented in (Bergomi, Paul, Solhaug, & Vignon-Davillier, 2013). While CORAS uses an import/export relationship between its artefacts, Rinforzando utilises dynamic links to connect them. A trace model in general needs to be able to specify traceability links between the target model and the risk model, to trace changes and to identify risks that need to be reassessed to preserve validity (Seehusen & Solhaug, 2012). However, traceability in risk assessment is proven to still be ill-defined, and further research is needed to formulate traceability in the domain of risk assessment to support a proper specification of a trace model.

3.3 Traceability definition

We investigated the notion of traceability in cyber risk assessment by referring to literature in other domains such as food industry and requirements engineering and by interviewing experts within TRE$_S$PASS to induce their opinions. This mixture of input led us to discover that there is an inherent duality as to what traceability entails in cyber risk assessment; one that we conveniently refer to as static and dynamic perspective. The static perspective construes that traceability should be able to support connection between input and output of a risk assessment process, and that it should be compatible with the mix of data types (such as stakeholder goals and IT infrastructure) used in the risk assessment process - among other aspects. The dynamic perspective of traceability chiefly deals with changes

that are influencing the system, such as the ability to flag changes and to keep a detailed log of changes. Based on this dichotomy, a formal definition for traceability in cyber risk assessment is prescribed as follows:

> In the realm of cyber risk assessment, traceability is the ability to show the rationales behind risks by generating bidirectional trace links spanning from the input to the risk treatment decisions, annotated with relevant and meaningful contextual information at each intermediate step. In its dynamic sense, traceability supports representation of relevant changes and its impact in the risk assessment process.

3.4 Previous traceability initiatives in TRESPASS

Based on this definition, possibilities for incorporating traceability in TRESPASS were identified. Existing initiatives in TRESPASS that involve traceability are the following:

- Washing instructions and patient information leaflets (The TRESPASS Project, D2.3.1, 2014), which describe how data for the models was obtained and how it can be used; this includes linking to scientific papers, vulnerability databases, expert judgements, etc., which are the source of data in the models;
- Callbacks in attack generation, in which additional information about agents involved in an action is obtained from the model; in order to be able to do this, the agents in the model that were the source of generating an attack action should be linked from the attack step.

In order to build traceability into security risk models, a more integrated approach needed to be developed. In the following, we will describe such an approach, based on lessons learnt from other traceability efforts, such as those in software development. This initiative contributes to evidence-based security, by making sources of modelling decisions explicit.

Within this effort, we have reviewed the TRESPASS integration diagram (The TRESPASS Project, D6.2.2, 2015) for other potential traceability challenges and solutions. In particular, we envisioned a role for the different kinds of libraries, in the sense of traceability of which patterns were used in which models and analyses. For example, a model template that was used as a basis for a model. Or an attack pattern that was used to refine a node in an attack tree. For traceability, it is essential to keep track of the patterns that were used, for example if changes occur in the patterns themselves.

3.5 The TRESPASS trace model

Departing from the definition above, the trace model was crafted on the basis of a three-step design science approach. In the first step, we investigate the problem by conducting a

stakeholder analysis for the trace model and formulate goals from the trace model. Goals that correlate with both the static view and the dynamic view of traceability were identified. These goals range from enabling tracing of data used in the analysis back to the respective data sources to supporting propagation of changes in the input data/model components throughout the analysis process. A social engineering attack scenario and a scenario of socio-technical changes in an organisation were used as illustration to aid the design process, taking into account the static view goals and dynamic view goals respectively. Both scenarios were formulated in the context of a case study on a private (on premise) cloud computing environment from TRESPASS.

The second step marks the actual design process, where we utilised the previously identified goals and results of interviewing the experts to elicit requirements for trace model. Another important source of requirements is Use Cases of traceability for users of TRESPASS. The Use Cases show how traceability can help users understand analysis results, rationalise risk treatment decisions, analyse impact of changes in the system, and conduct sensitivity analyses in TRESPASS. A total of 31 requirements for the trace model were formulated, which were then refined into design criteria and powdered into structural specifications. Our specifications take assorted forms: metadata to support data management process, language embeddings, and functionalities to append comments to the model and analysis results, to name a few.

In the third and last step, we (1) check whether the trace model has fulfilled its every requirement and (2) study its behaviour in a different context from the one in which it was designed. These were done in a set of activities collectively referred to as validation. To achieve the former goal, verification sessions with a similar group of experts were conducted. With the latter goal, we run select Use Cases in a single-case mechanism experiment; this time with a case study on outsourced cloud computing environment. From the verification sessions we learned how viable our specifications are for implementation, whether there are restrictions such as past decisions in the project that could hinder implementation, and whether our specifications are in line with the TRESPASS tools as they are being developed. A list of validated specifications for the trace model were then formulated (Table 3.1).

Table 3.1: Summary of validated specifications

Code	Original specification	Modification	Source of validation
SP2	Metadata `data_domain` and `data_type` for model components.	(+) Annotate with explanation of the fields	O
SP3	Metadata `data_source`, `created_by`, and `last_updated_by` for model components.	(+) Metadata `date_of_origin` (+) Annotate with explanation of the fields	O
SP4	Storage of model components metadata in knowledge base.	(no change)	-

Table 3.1 Summary of validated specifications (continued)

Code	Original specification	Modification	Source of validation
SP5	Metadata `measurement_type` for model components.	(+) Metadata `aggregation_level` (+) Annotate with explanation of the fields and make the `measurement_type` field open	O
SP6	Metadata `created_on`, `last_updated_on`, and `status` for model components.	(+) Annotate with explanation of the `Status` field	-
SP7	Functionality to import manual model during model creation.	(+) Feature to upload minutes of meeting	O
SP8	Functionality to tag pictures during model creation.	(no change)	-
SP9	Metadata `templatemodel`, `genericattacker`, or `genericentity` for generic elements or templates.	(+) Values *No* and *Modified* for the metadata	O
SP10	Sub-wizard **Comments** during model creation and functionality to add comments to model components.	(no change)	-
SP11	Unique label for parameters in APLTool.	Create format for node IDs (`APLxxx`) and attach them to the parameters	O,E
SP12	Parameters for augmentation and annotation are pushed back to the knowledge back for storage.	(-) already implemented	E
SP15	Linkage of model components in XML format to attack tree nodes.	(-) already implemented	O,E
SP16	Functionality to show metadata fields of model components in attack tree nodes.	(-) duplication with SP24	O
SP17	Metadata `data_id` for model components.	(-) parameters are class-specific, not instance-specific	O
SP18	Metadata `id` for attacker profiles.	(-) already implemented	O,E
SP19	Metadata `timestamp`, `origin`, and `version` for data imported into the knowledge base.	(+) Metadata to indicate sharing permission `Permission`. Possible values: *Restricted* and *Public*.	O
SP20	Linkage of Actors or Assets to unique parameter labels in APLTool.	(-) already implemented	O
SP22	Metadata `node id` for nodes generated in APLTool.	Create format for node IDs (`APLxxx`)	O
SP23	Linkage of `parameter name` defined in APLTool to analysis tools.	(-) already implemented	O
SP24	Functionality to show `node id` in visualisation results.	(+) show parameters of model components and calculation of values	O

Table 3.1 Summary of validated specifications (continued)

Code	Original specification	Modification	Source of validation
SP25	Functionality for global search of entity.	Results to be shown visually, not in a list.	O
SP26	Sub-wizard **Comments** in visualisation interface for general comments and functionality to attach comments to individual nodes visualised.	(+) Heading to categorise comments.	O,E
SP27	Storage of comments in navigator map in .txt format.	Change format to rich text instead of plain text.	O
SP28	Push notifications of updates from external data sources or automated data extraction tools.	Start incrementally by first writing script for automated update, then proceed to developing user interface.	O
SP29	Functionality to generate summary of changes made to the model.	(+) Add warning that changes have been made	O
SP30	Metadata `timestamp`, `owner`, and `version` for analysis results.	(-) Delete `owner`. TRESPASS does not recognise different users.	O
SP31	Functionality for comparison of different analysis results file.	(+) Comparison of countermeasures and their costs, and comparison of visualisations.	O

Note: *O*: Opinions; *E*: Experiment

The final features of the trace model are summarised in an overarching figure (Figure 3.1).

Besides the formal definition, our contributions to state of the art on risk assessment include the application of metadata not only to provide more context during Context Establishment, Risk Identification, and Risk Analysis activities, but also the context of the entire risk assessment exercise; the demonstration of building linkage between context establishment stage and risk identification stage via identifying the entities involved in the manual risk model; and the application of text-based explanation to support linkage between context establishment and risk identification activities, as well as risk evaluation results and risk treatment decisions. In formulating these features, past decisions and ongoing developments made within the project were accounted for.

Improvements such as integration with existing metadata standards and structuring text-based explanation will complement the trace model as it stands now. We believe our ideas for traceability are relevant with research initiatives around the theme of digital security as put forward by Horizon 2020. With the ever-increasing cyber dependency trend, organisations should no longer treat traceability in risk assessment as an option. Its role will grow to become more important to help maintain a robust risk assessment.

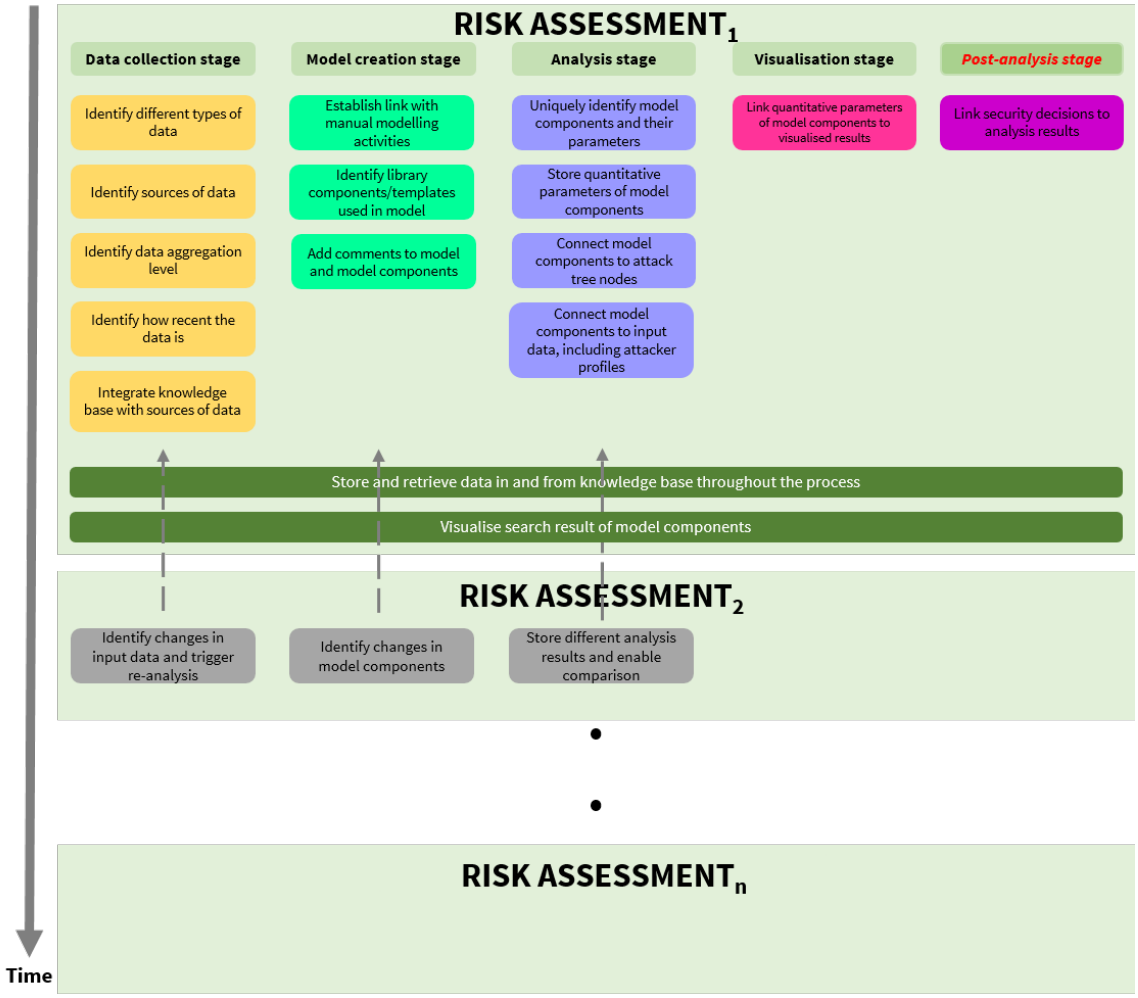

Figure 3.1: The TRESPASS traceability model

3.6 Conclusion

This chapter provided an extensive analysis of the role of traceability in TRESPASS. Based on an analysis of traceability in other domains, it provided requirements for a trace model, highlighting both existing best practices in the project with respect to traceability, as well as opportunities for further embedding traceability both in the different stages of the TRESPASS process, and in cyber risk management in general.

4 Model Transformation

Model transformations, more specifically, graph transformations, provide a generic and flexible modeling formalism (Mens & Van Gorp, 2006; Ghamarian, de Mol, Rensink, Zambon, & Zimakova, 2012). This formalism is widely used in model-driven software engineering. Generally, model transformation is a process which takes one or more input (source) models and produces one or more output (target) models (Sendall & Kozaczynski, 2003). This process can enable different views on the same system (horizontal transformation), or it can be applied to move the model to a different level of abstraction (vertical transformation). Furthermore, model transformation can be applied to ensure consistency and coordination between different models (Sendall & Kozaczynski, 2003).

The TRE$_S$PASS process applies state-of-art model transformations at numerous steps. In this chapter we summarize the main successful application results of model transformations in the project and give pointers to project's deliverables and papers that report these achievements. Besides these pointers, Deliverable D3.5.1 *Dynamics of Stochastic Models* (The TRE$_S$PASS Project, D3.5.1, 2016) reports more details on the model transformation techniques applied in TRE$_S$PASS.

4.1 Horizontal model transformations

The main application of horizontal model transformation in TRE$_S$PASS is the Treemaker tool that creates automatically an attack tree from a socio-technical system model and a chosen high-level attack scenario (comprising some information about the attacker and the asset he/she wants to attack) (The TRE$_S$PASS Project, D3.4.1, 2014; The TRE$_S$PASS Project, D3.4.2, 2016; Ivanova, Probst, Hansen, & Kammuller, 2015). The approach implemented in the Treemaker applies the reachability reasoning to the elements in the socio-technical model. More precisely, it reasons about how elements of the socio-technical model can be *reached* (accessed, moved to, etc.) based on the links available in this model. Therefore, the approach transforms the connectedness view of the socio-technical model into the precise attack path view in an attack tree. Furthermore, in TRE$_S$PASS we have extended this approach for generating also attack-defense trees. This formalism combines simultaneously views of the attacker (what is the attack paths available) and the defender (what are the defensive mechanisms that block potential attack paths) (The TRE$_S$PASS Project, D3.4.1, 2014; The TRE$_S$PASS Project, D3.4.2, 2016; Gadyatskaya, 2015).

4.2 Vertical model transformations

The vertical model transformation paradigm is pertinent to the TRESPASS approach in general, as it follows the *navigation metaphor*, i.e., the analyst typically starts from the satellite (high-level) view of the system and subsequently refines it into a more fine-grained socio-technical system model, or a series of such more fine-grained models (Pieters, Barendse, Ford, Heath, & Probst, 2016; Probst, Willemson, & Pieters, 2016; The TRESPASS Project, D5.4.2, 2016). The high-level view of the system is created by the analyst together with the stakeholders during, e.g., a LEGO modelling exercise, and, subsequently, a more refined model is produced in the Attack Navigator Map (ANM) (The TRESPASS Project, D4.3.3, 2016). The project has devised an app to automatically pull the details from a LEGO prototype into the formal socio-technical model expressed via the ANM (The TRESPASS Project, D4.3.3, 2016). In this way the important details of the original high-level system view are preserved automatically.

Another application of vertical model transformation techniques in TRESPASS consists in automatic augmentation of a generated attack tree with sub-trees from the Treebank (the attack pattern library) and data from the knowledge base (The TRESPASS Project, D2.4.1, 2016; The TRESPASS Project, D5.4.2, 2016). In this step the ANM creates a more refined attack tree that is ready for quantitative analysis.

Another application of vertical model transformation in TRESPASS is the transformation of attack trees into more refined attack models such as timed automata and stochastic Markov chains (The TRESPASS Project, D3.5.1, 2016; Jhawar, Lounis, & Mauw, 2016; Gadyatskaya, Hansen, et al., 2016). This approach allows to capture temporal and causal dependencies among events that are typically ignored by the traditionally static attack trees. Thus, these more refined attack models are quite useful for capturing continuous dynamics of the system. We provide further details about this project's result in Section 6.6 as it is also relevant to practical maintenance of attack models.

4.3 Coordination between models

TRESPASS has proposed an attack tree meta-model to allow bridging individual analysis techniques for many (existing and future) dialects of the attack tree language (The TRESPASS Project, D3.5.1, 2016; Huistra, 2016). The meta-model enables mutual interoperability of several analysis tools (ADTool, ATTop, Attack Tree Analyzer, Attack Tree Evaluator) and the Treemaker. Furthermore, the meta-model was useful for enabling the attack tree models translation into the more refined timed automata model of UPPAAL (The TRESPASS Project, D3.5.1, 2016).

4.4 Conclusions

As we have seen, model transformation techniques are widely used in TRE$_S$PASS. These techniques allow the analyst to better manage the complexity of system descriptions, and support the interplay between different analysis tools. Moreover, the essential for the TRE$_S$PASS process step of attack tree generation from a socio-technical model can also be interpreted as a model transformation technique. Furthermore, it is evident that model transformations play a critical role in the model maintenance within the TRE$_S$PASS tool portfolio. They facilitate dealing with emerging security scenarios (if a socio-technical model is updated then the corresponding attack scenarios will be updated as well); and they support model sharing activities, as different analysis tools can share their input models through the attack tree meta-model.

5 TRESPASS View on Dynamic Situations

The questions of dynamic model aspects and maintaining analyzability of models have been extensively studied in the project. In this chapter we briefly review the relevant project results and give pointers to the relevant deliverables and papers.

5.1 Socio-technical model dynamics

One approach to treat analyzability of models in dynamic situations and to ensure continuous exploitation of the modelling paradigms introduced is to introduce dynamic aspects, e.g., context-triggered changes of behaviour, in the model itself. This approach was investigated for socio-technical models in D1.3.3 *Dynamic Features of Socio-Technical Models* (The TRESPASS Project, D1.3.3, 2015). In this deliverable the TRESPASS consortium has systematically studied dynamic features useful for socio-technical security models based on the case study requirements and the socio-technical threat evolution.

The main finding of D1.3.3 with respect to the dynamic aspect requirements imposed by the case studies is that the TRESPASS modelling language is sufficiently rich to be able to cover the vast majority of the necessary dynamic features. We provide more details on the socio-technical model maintenance and dynamic aspects in Chapter 6. The non-covered aspects can be further introduced by extending the modelling language (see (The TRESPASS Project, D1.3.2, 2015) for more details on extending the socio-technical model), or by enhancing the models with dynamic sources of information (for instance, up-to-date information on organizational hierarchy and applicable regulations). Deliverable D1.3.4 *TRESPASS Socio-Technical Model and Application Languages* reports on the final version of the TRESPASS socio-technical model and covers extensively such dynamic modelling aspects as policies and processes (The TRESPASS Project, D1.3.4, 2016).

We further refer the reader to the Deliverables D2.2.2 *Data Extraction from Virtualized Infrastructures* (The TRESPASS Project, D2.2.2, 2015), D2.2.3 *TRESPASS Technical Data Extraction Tools* (The TRESPASS Project, D2.2.3, 2016), and D2.3.2 *TRESPASS Social Data and Policy Extraction Techniques* (The TRESPASS Project, D2.3.2, 2015) for more information about best practices and techniques for handling data extraction for the TRESPASS process purposes. Deliverable D2.4.1 *TRESPASS Information System* (The TRESPASS Project, D2.4.1, 2016) reports on the TRESPASS knowledge base that handles the extracted data and stores it for future use and sharing. From the TRESPASS process perspective, the project has also investigated dynamic risk assessment as a way of handling dynamicity (see Chapter 5 of this deliverable).

To illustrate the required synergy of dynamic features of the socio-technical model with dynamic data handling, in the next section we summarize the dynamic challenges pertinent to the cloud case study, as an example of a case study with strong requirements on dynamicity.

5.2 Model dynamics in the cloud

Cloud computing has gained remarkable popularity in recent years due to the economic and technical advantages of this new way of delivering computing resources. Customers benefit from rapid provisioning and seemingly infinite scalability, while only being charged on a pay-per-use basis. In order to fulfil these promises, cloud infrastructures are implemented as *Software Defined Environments* (SDEs): more and more adjustments traditionally requiring hardware and cabling changes, and implemented on the timescale of days, can now occur in the virtual environment by just *redefining* the virtual environment. This leads to a high dynamism: within seconds, virtual servers can be created, destroyed or moved, with corresponding network changes and reconfigurations, but also new user ids can quickly be introduced, potentially with high-level access, and removed within the blink of an eye. Additionally, the cloud service provider itself, respectively its administrators and technicians, are new actors that have a very high level of access and control over the IT infrastructure that is typically shared across many companies and users in case of a public cloud offering, leading to new risks to be considered.

Although this leads to a highly complex and quickly changing environment, an advantage of the software-defined nature is that all components are managed in a consistent way, usually from one or few central components. This makes it possible to get very rich, highly detailed, timely and accurate information about the infrastructure and enables an automated extraction of the setup and status of the virtualised environment to a high level of detail (see (The TRE$_S$PASS Project, D2.2.2, 2015; The TRE$_S$PASS Project, D2.2.3, 2016)).

Similarly, it is possible that the management component triggers further processing on selected or even every change in the virtualised environment (Bleikertz, Vogel, & Groß, 2014). Not all changes in the socio-technical model happen fast though: physical changes and social changes are slow compared to the changes in the virtualised environment.

One important aspect of the cloud scenario within TRE$_S$PASS is therefore the need to investigate the practices and trade-offs of handling the risks on different timescales. For example, checking/enforcing compliance rules (e.g., “no network path may exist between certain network domains”) has to take place on all timescales, whereas support of risk analysis and planning including social and physical elements, like the value of introducing a new authentication mechanism for system administrators, will entail completely different timescales and will not require all changes to happen at the level of the virtual infrastructure.

Where one wants or needs to follow these changes in real-time, processes in the tool chain from data collection to analysis results need to be aware of this dynamic nature and

must be triggered by changes and ideally happen in an incremental fashion, due to constraints in the volume of data and time available for processing. Deliverable D7.2.2 (The TRESPASS Project, D7.2.2, 2016) reports on the cloud case study results and evaluates, amongst other features, how the TRESPASS process and tools handle the requirements for a dynamic environment.

5.3 Compositionality of analysis

Last but not least, task T3.5 *Dynamics of Stochastic Models* in WP3 is dedicated to ensuring that the analysis process is able to cope with dynamic socio-technical models and with the constant presence of change in the organisations being modelled. To achieve this goal, TRESPASS has investigated a compositional analysis process that imposes less computational requirements and thus facilitates maintaining analyzability of models for changing situations. Moreover, these models can capture dynamic attack scenarios, unlike attack trees, which is a static formalism (The TRESPASS Project, D3.4.2, 2016). More details about the compositional analysis for stochastic attack models is provided in Sec. 6.6 of this deliverable, and in (The TRESPASS Project, D3.5.1, 2016).

5.4 Conclusions

As we have seen in this chapter, TRESPASS has produced many techniques and tools to tackle dynamic environments and changing situations. We have approached the challenges arising from the dynamicity requirements at every step of the TRESPASS process (The TRESPASS Project, D5.4.2, 2016). The project has designed the socio-technical model that can handle dynamic elements (The TRESPASS Project, D1.3.4, 2016; The TRESPASS Project, D1.3.3, 2015); and it has developed a knowledge base that is able to fulfil the requirements on the rapidly changing nature of the data (The TRESPASS Project, D2.4.1, 2016). The project has also tackled quantitative analysis on dynamic attack models (The TRESPASS Project, D3.5.1, 2016). Moreover, we have provided the means to visualize complex, evolving information to the analyst (The TRESPASS Project, D4.3.2, 2016; The TRESPASS Project, D4.2.2, 2016). We can conclude that the TRESPASS process comprising these elements allows us to deal holistically with dynamic situations.

6 Technical View on Model Maintenance

In this chapter we overview individually the main models used in the TRESPASS process and summarize how these models are maintained in changing situations. Here, by a *changing situation* we consider scenarios if the organization evolves, its systems change, or new information about attackers or attacks becomes available, i.e., both internal and external changes for the company.

6.1 Socio-technical model maintenance

Socio-technical models represent the TRESPASS process's knowledge about the infrastructure of an organisation on the physical, virtual (digital), and social levels. Usage of such models in security risk assessment is one of the key innovations introduced by TRESPASS, because these models enable automated reasoning about the security posture of the organization (The TRESPASS Project, D1.3.4, 2016).

All elements of a socio-technical model have *unique identifiers* that can be used to attach parameters and properties to the elements. Maintenance of the model of the organization can be expressed as a change in the model itself, representing changes in the infrastructure, but also as a change in the data associated with model elements.

Model maintenance can occur either as a manual process by editing the model directly or through a GUI (the Attack Navigator Map, ANM (The TRESPASS Project, D6.4.4, 2016)), or through automatic extraction of data from the organisation, e.g., in the case of data extraction from cloud infrastructures. Deliverable D1.3.4 *TRESPASS Socio-Technical Security Model and Specification Languages* (The TRESPASS Project, D1.3.4, 2016) presents the final version of the TRESPASS model formalism and defines the means (the language) to specify organizational models. Deliverable D6.4.4 *The Integrated TRESPASS Tools* (The TRESPASS Project, D6.4.4, 2016) presents the implementation of the TRESPASS model in the ANM and overviews the important design decisions to maintain instantiations (versions) of organizational models.

To record historic changes in models, the TRESPASS knowledge base (The TRESPASS Project, D2.4.1, 2016) saves history information of models, which is accessible through the ANM. This approach takes snapshots of the model at any time the TRESPASS process starts another iteration, such ensuring that infrastructure information and values are recorded.

Model maintenance is further tackled in deliverables D1.3.3 *Dynamic features of socio-technical security models* (The TRESPASS Project, D1.3.3, 2015), and D1.3.2 *Extensibility of socio-technical security models* (The TRESPASS Project, D1.3.2, 2015). Dynamic features mostly address how the model is able to deal with changes in the organisation and threat landscape; these aspects are the triggers for model maintenance, as they might require a change to the model itself, or even might require changes to the modelling language, the tools, and/or the TRESPASS process itself. Support for these changes at the modelling language level is described in (The TRESPASS Project, D1.3.2, 2015).

Changes to the model, tools, data, or parts of the process must trigger re-evaluation of analyses, attack generation, visualisations, etc. These changes are captured methodologically in the TRESPASS process as a loop (see Chapter 2 of (The TRESPASS Project, D5.4.2, 2016)).

6.2 Pattern model maintenance

Attack patterns are attack (sub-)trees where the leaves are annotated with quantitative parameters like cost, success likelihood, difficulty, and time. The attack tree generated by the Treemaker is augmented with patterns from attack pattern library and the knowledge base to increase modeling granularity (The TRESPASS Project, D5.4.2, 2016; The TRESPASS Project, D2.4.1, 2016).

The patterns will change when an update is provided to the attack pattern library. Such an update is rather straightforward and does not require any extra steps. Indeed, it is expected that the analyst will continuously populate the attack patter library, and will ensure that it is up-to-date (The TRESPASS Project, D5.4.2, 2016).

6.3 Attack-defence trees maintenance

Attack-defence trees are a security model that combines the views of an attacker and a defender on a situation (Kordy, Mauw, Radomirović, & Schweitzer, 2011, 2012). Intuitively, one may view attack-defence trees as attack trees augmented with countermeasures. However, attack-defence trees may also be considered as countermeasure trees augmented with attacks. In TRESPASS attack-defence trees have become a model of choice for formalisation of countermeasure selection based on attack trees (The TRESPASS Project, D3.4.2, 2016; Gadyatskaya, Harpes, Mauw, Muller, & Muller, 2016). We also envisage that once an attack tree is generated from the socio-technical model, annotated with data, pruned and augmented with the information from the attack pattern library and the knowledge base, the analysis process will proceed to evaluate quantitative questions asked by the user (e.g., what is the probability of attack given certain attacker's budget) and suggests some countermeasures that can mitigate the most dangerous attacks. These countermeasures can be introduced directly in the socio-technical model. In this case, the process of attack generation can start again (The TRESPASS

Project, D3.4.2, 2016). Alternatively, TRESPASS has created a tool ADTop that can directly suggest countermeasures based on a cost-benefit analysis and incorporate them directly into attack tree in order to continue the analysis on the resulting attack-defence tree (Gadyatskaya, Harpes, et al., 2016; The TRESPASS Project, D3.4.2, 2016). Moreover, attack-defence trees can be generated directly from the socio-technical model (The TRESPASS Project, D3.4.2, 2016; Gadyatskaya, 2015). Thus, we have considered how to maintain attack-defence tree models produced during the analysis process.

We consider the following situations in which attack-defence models need to be updated:

- **Consistency maintenance (with respect to the underlying socio-technical model):** Socio-technical model is updated due to some structural changes in the modelled organization. In this case the attack-defence tree created in the analysis process need to be brought to consistency with the socio-technical model. Straightforwardly, this is done by repeating the analysis process and regenerating the attack-defence tree model (from the socio-technical model directly, or by first regenerating the underlying attack tree model and then enhancing it with countermeasures).
- **Completeness and soundness maintenance (with respect to countermeasures):** We envisage that a library of countermeasures exists that provides the analyst with some choices of countermeasures, and the relevant data on their cost and effectiveness (The TRESPASS Project, D3.4.2, 2016; Gadyatskaya, Harpes, et al., 2016). In this case, if this library is updated, then the attack-defence tree models need to be updated too, because they might not include the most suitable countermeasures, or might include countermeasures that are now obsolete. In this case attack-defence tree regeneration can be an option again, similarly to the previous case.

While currently not implemented in the TRESPASS process, the attack-defence tree models could be updated not by repeating the full analysis, what can be time-consuming, but based on the traceability links established among the TRESPASS models. More information on the traceability link maintenance study of TRESPASS is given in Chapter 3.

6.4 ArchiMate enterprise architecture model maintenance

Within TRESPASS, the ArchiMate enterprise architecture modelling language (The Open Group, 2016) is used in a number of ways, e.g.:

- As a user-friendly “front-end” to model the as-is infrastructure, from which a socio-technical security model can be generated.
- As a design language to model the high-level design of the to-be infrastructure, including the required control measures.
- To model the vulnerabilities, threats and risks identified in a risk analysis, by means of the proposed risk and security extension described in a white paper of The Open Group (Band et al., 2015).

In the context of the first two items, ArchiMate is an exemplar of an architecture or design language, comparable to UML as a detailed design language for IT systems and technical infrastructure, and BPMN as a detailed design language for business processes. Therefore, many of the maintenance issues of this type of languages in general also apply to ArchiMate models.

Section 3.2 of the TRE$_S$PASS deliverable on the extensibility of socio-technical security models (The TRE$_S$PASS Project, D1.3.2, 2015) provides some more detail on how the ArchiMate language can be extended to incorporate risk and security modelling and analysis. This section also includes an example of how the ArchiMate language has been applied in the IPTV case study.

Traceability between model elements is an important mechanism that helps to maintain consistency, both between elements within the same model (e.g., requirements and parts of the architecture that realize these requirements, both part of the same ArchiMate model) and between different related models (e.g., an infrastructure model in ArchiMate and the TRE$_S$PASS socio-technical security model generated from this). Also traceability between model data and other types of data may play a role here, e.g., input data gathered by the TRE$_S$PASS data collection tools or analysis results generated by the analysis tools. Modelling tools play an important role in providing this traceability.

In ArchiMate, the *viewpoint mechanism* also helps to improve the maintainability of models. When the same model element appears in multiple model diagrams, rather than using separate copies of the element, each of the diagrams has a view reference to a single underlying model element. In this way, the name and other relevant properties are always kept consistent. Again, modelling tools play an essential role in implementing this mechanism.

Version control of models is a prerequisite for model maintenance, making it possible to track changes to a model over time and, if necessary, revert to earlier versions of the model. Most of the commercial ArchiMate modelling tools can be used in combination with a model repository that supports version control.

6.5 e3value value model maintenance

Value models describe the high-level interaction between profit-loss responsible actors. Unlike process models and activity diagrams they contain no information about ordering or timing. Unlike lower-level (enterprise) architecture, they do not relate to any implementation-specific details, such as networks, systems, data or logic. The value modelling approach of choice in the TRE$_S$PASS project is the *e3value* ontology and toolkit[1] (The TRE$_S$PASS Project, D7.3.2, 2016).

The *e3value* methodology provides modeling concepts for showing which parties exchange things of economic value with whom, and expect what in return (Gordijn, Akkermans, & Van Vliet, 2000). As such, *e3value* models describe actors and exchanges of

[1] http://e3value.few.vu.nl/

value (be it products, services, knowledge or other). We use *e3value* models for assessing the financial risk associated with the fraudulent misuse of a service package, in what we call the *e3fraud* approach(Ionita, Wieringa, Wolos, Gordijn, & Pieters, 2015): A value model of the service package under assessment is constructed and using TRE$_S$PASS tools, many potential "sub-ideal" models are automatically generated and ranked. The e3fraud approach is positioned within the larger TRE$_S$PASS workflow in (The TRE$_S$PASS Project, D5.4.2, 2016) and is applied to several telecom case studies in (The TRE$_S$PASS Project, D7.3.2, 2016), while details on generation and ranking can be found in (The TRE$_S$PASS Project, D3.4.2, 2016).

As value models describe a service package, the resulting models pertain to a high-level or reusability in sectors such as Telecom, where most services packages are made up of standard actors - such as a caller, callee, home provider and remote provider - performing common actions - such as calling or using data - and engage is recurring payments - such as monthly subscriptions and interconnection fees -. A library of "building blocks" (essentially partial models descrying atomic services with no pricing or usage information attached yet) could be maintained in order to further reduce modelling effort. Minor maintenance of these models will suffice, as they would have to be instantiated on usage anyway.

Furthermore, since there is no formal relationship between the other TRE$_S$PASS models and *e3value* models, maintaining consistency is not an issue.

6.6 Lower order attack model maintenance

Traditionally attack trees models are static. In order to incorporate temporal and causal dependencies, attack trees need to be reduced to lower level models, such as transition diagrams. These lower level models – Timed automata (The TRE$_S$PASS Project, D3.4.2, 2016; The TRE$_S$PASS Project, D3.5.1, 2016; Gadyatskaya, Hansen, et al., 2016), stochastic Markov chains (Jhawar et al., 2016), I/O interactive Markov chains, etc. – provide a clear semantics to the behavioural properties of higher-level formalisms (the socio-technical model or an attack domain-specific language (DSL), such as attack trees or attack-defence trees). Thus, they are quite useful to model the real-time continuous dynamics, capturing the system's non-deterministic behaviour. Furthermore, they are compositional.

Compositionality is important in socio-security scenarios as the considered scenarios are complex and constructing and analysing them directly usually suffers from state space explosion, necessitating abstractions and thus making models imprecise. It gives us the flexibility of transforming attacks into more lower level formalisms (automata, or Markov chains), which can then be used as a sub-system in a larger system. This keeps the model modular, flexible and easy to extend (Boudali, Crouzen, & Stoelinga, 2010). Thus, even if there is an additional sub-system expressed as a sub-attack tree, all we need to do is to translate it into a transition diagram and use it in the larger system.

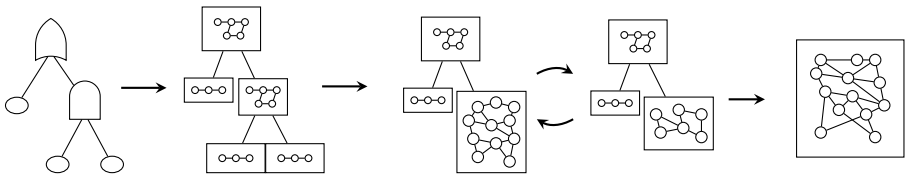

Figure 6.1: Graphical overview of compositional aggregation for AT models.

Instead of generating one aggregated model from an attack tree, we translate basic attack steps (BASs) into respective transition diagrams, which synchronise on their action labels. Thus, instead of composing the whole tree at once, we compose smaller sub-trees in a stepwise fashion and then minimise the state space after each composition using notions of strong, weak and branching bisimulation, as studied in the process algebra. Therefore, we circumvent the omnipresent state space explosion problem by not constructing a large stochastic model in the first place. A graphical representation of the approach is shown in Figure 6.1. More details about this compositional approach that facilitates maintenance of large attack models are provided in (The TRE$_S$PASS Project, D3.4.2, 2016; The TRE$_S$PASS Project, D3.5.1, 2016).

6.7 Timed automata model maintenance

Timed automata (TA) are a well-known formalism that has been used to model and analyse a wide variety of systems, e.g., attacks in socio-technical systems, communication protocols, safety-critical applications etc. In TRE$_S$PASS timed automata models are either generated automatically from attack trees (see (The TRE$_S$PASS Project, D3.5.1, 2016; The TRE$_S$PASS Project, D3.4.2, 2016)), or created manually by directly modelling the (socio-technical) system of interest (David et al., 2015; Gadyatskaya, Hansen, et al., 2016). The primary advantage of manual modelling is that it allows for the full expressive power of timed automata to be used, resulting in better/more precise models, and thus analyses.

Maintenance of a manually created TA model is a similarly manual process that is highly dependent on how the model was initially created and what features of the modelling language that has been used, e.g., whether or not the original model has been created in a systematic and modular manner (with a low degree of interdependence in the model) or in a more ad-hoc manner. Furthermore, some analysis tools, e.g., UPPAAL, support models to be developed in networks or hierarchies that are inherently (more) modular.

6.8 Data and model versions maintenance

The TRESPASS information system serves as an integrator for the ANM, the data collecting methods and the analysis tools (The TRESPASS Project, D2.4.1, 2016). This system serves as a repository for data, patterns, and models created in the TRESPASS process. To allow efficient operations with these items, especially from the maintenance perspective, the project has chosen to implement the knowledge base as a versioning system akin to git, rather than a traditional database relying on relational data schemes. This choice enables us to profit from iterative, progressive modeling and refinement for the security scenario at hand, and it supports subsequent reviews and audit of risk assessment process instances.

6.9 Conclusions

In this chapter we have discussed various models used in the TRESPASS analysis process and how these can be maintained when the organisation evolves or the analysis needs change. TRESPASS follows a modular approach, as for all models we have developed means to maintain their analyzability in presence of a change. Furthermore, as the TRESPASS relies substantially on model transformations (see Chapter 4), there is an automatic approach to maintain analyzability of interconnected models (by propagating the changes via transformations).

7 Dealing with New Situations in Risk Assessment

In Chapter 6 we discussed how the underlying TRESPASS models can evolve and how to maintain the validity of analysis in the presence of change. Another important aspect of change that we have to consider concerns changes in the environment that affect the data and estimations used in the TRESPASS analysis process. TRESPASS has designed means to automatically pull new data from the environment in the context of the cloud case study (The TRESPASS Project, D2.2.2, 2015). However, not all infrastructures allow for quick data collection. To assure that the TRESPASS risk assessment process still produces valid outputs, we have further considered dynamic risk analysis. In cooperation with the research project IDS4ICS[1] by itrust consulting, we have assessed how to deal with new security situations in risk assessment.

Indeed, risk assessments constitute an important tool for determining the risk that comes with a project or an activity. The outcome of such an analysis is generally used to take reasonable decisions and implement appropriate countermeasures against identified threats. Although they are commonly designed for a one-time use or inspection, risk assessments have much greater potential.

For instance, a dynamic risk analysis, which would be continuously re-evaluated, permits the inspection of the real-time risk that a project or activity is facing right now. As a consequence, it could serve as a basis for everyday decision making. To a certain extent, notably in the context of intrusion detection and prevention, it could even enable the system to take the decisions by itself. In fact, the additional knowledge of the actual value of the affected assets, or the real impact of a threat, constitutes undoubtedly an added-value for dynamic risk analyses, compared to traditional monitoring or detection utilities. Indeed, a dynamic risk analysis aids in recognizing the overall risk exposure, identifying critical assets and thus, prioritizing incidents.

In traditional risk assessments, all parameters have to be estimated manually, including the likelihood that a scenario occurs, the impact of a threat or the efficacy of a countermeasure. Note that for technical assets, one often has the possibility to roughly 'measure' such a parameter – it does not have to be very precise, since risk assessments are supposed to be insensitive to fluctuations, so that an estimate of the order of magnitude should be enough.

Examples of such parameters include the health of a physical disk using the so-called S.M.A.R.T. system (likelihood of disk failure), the number of servers exposed to the internet

[1] http://fnr.lu/projects/risk-monitoring-with-intrusion-detection-for-industrial-control-systems-2/

(impact of distributed denial-of-service), or the percentage of machines protected by anti-virus (counter-measure implementation rate).

Even if a parameter cannot be fully estimated, one can often still encode a certain dependency on measurable factors.

The proposed model of a dynamic risk analysis is based on the following concept. Every measurable factor or parameter constitutes a variable in the risk analysis tool. For every such variable there is a responsible probe hosted on the respective system, which continuously reports the current value of the variable.

As a consequence, the risk assessment is no longer defined by the parameter values, but by a function of these parameters, whereas the parameter values are deduced by the risk analysis tool itself. Formally speaking, the static risk (of asset i) is computed as

$$\text{risk}_i = \text{impact}_i \times \text{likelihood}_i \times \text{vulnerability}_i,$$

where impact_i, likelihood_i and vulnerability_i are estimated numbers.

In contrast, the dynamic risk is defined to be

$$\text{risk}_i = \text{impact}_i(\mathcal{P}) \times \text{likelihood}_i(\mathcal{P}) \times \text{vulnerability}_i(\mathcal{P}),$$

where $\text{impact}_i(\cdot)$, $\text{likelihood}_i(\cdot)$ and $\text{vulnerability}_i(\cdot)$ are estimated functions that depend on the set $\mathcal{P}$ of measured parameters.

There are several ways to deal with the variables that are reported to the central risk analysis tool.

- "Every time a probe reports a value, make it the new value of the associated variable until a new one is reported." This is reasonable in conjunction with static measurements or performance ratings, such as the implementation rate of a counter-measure (e.g. % of machines with enabled anti-virus solution).
- "Every time a probe reports a value, set the variable to that value, but make it decrease with time." Such a rule is useful in the context of incident reporting where each single alert raises the threat level to a certain value, but where there is no 'all-clear' signal that would lower the threat level over time. Malware detection constitutes a typical example for this category: if malware has been found in the network, there is a certain probability that computers have been affected.
- "Every time a probe sends an alert, increase the variable value a bit (depending on the severity of the alert), but make it decrease with time." This behaviour is expected with detection systems where a collection of evidence increases the overall disaster probability. This includes most notably intrusion detection systems, where alerts do not necessarily imply break-ins, but rather hint that there is possibly an intruder in the network.

A prototype has been implemented in TRICK Service (a security risk management system developed by itrust consulting) as a proof-of-concept on real-time risk monitoring as shown in Figure 7.1. The prototype has been validated with the ÉpStan TTP risk analysis which deals with a web service hosting personal data. More details about this case study and the evaluation results are available in (The TRESPASS Project, D7.2.2, 2016).

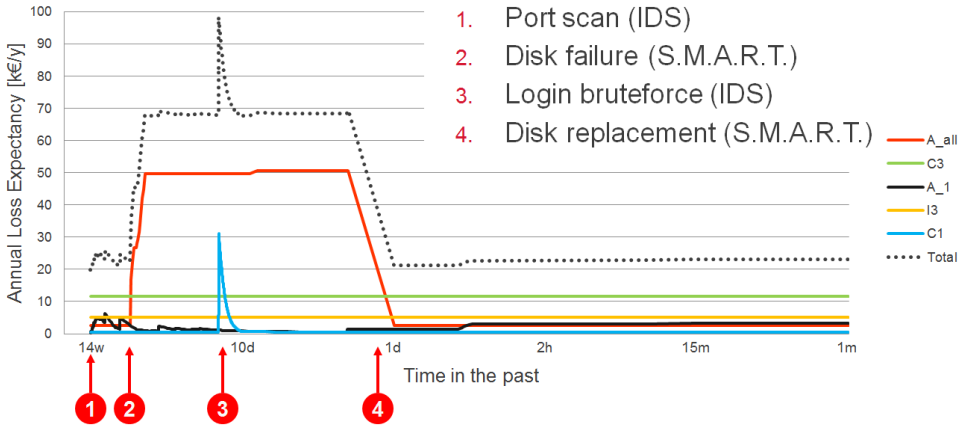

Figure 7.1: Real-time risk monitoring.

Conclusions. Dynamic risk assessment can be very useful for organisations in everyday security decision making as it tracks relevant changes in the organisational environment. We have applied dynamic risk assessment techniques implemented to the ÉpStan case study, and we can conclude that it is a useful methodology that can be further enhanced to the scale of the full TRESPASS process as a part of subsequent exploitation activities.

8 TRESPASS Best Practices

In this chapter we summarize the best practices introduced by TRESPASS for model maintenance and give guidelines on selecting the relevant techniques for model maintenance for users who are working on similar security tool ecosystems.

8.1 Requirements for model maintenance

As stated previously, we have worked to address two important aspects of model maintenace:

- *Continuous maintenance of complex models.* This aspect required techniques and methods to enable effective knowledge building, knowledge sharing and to generally make model maintenance more straightforward and usable for the analyst.
- *Facilitation of model maintenance in the presence of change.* This aspect demanded tools to ensure that the whole TRESPASS process is agile, i.e., it is possible to accommodate emerging security risk scenarios and new information at all stages of the TRESPASS process.

These two aspects were identified by analyzing the organic needs of the socio-technical risk assessment process of TRESPASS, by identifying the state-of-practice in the security industry, and by studying the state-of-art in model-driven engineering and legacy of relevant EU research projects (as reported in Chapters 1-4). Once we had pinpointed these goals as the most important ones for model maintenance in TRESPASS, we started to work towards providing the means to achieve them.

8.2 Summary of model maintenance techniques

Figure 8.1 conceptualizes, in context of the TRESPASS process, some core techniques developed by TRESPASS for model maintenance in order to achieve the two goals stated above. It demonstrates that model transformation techniques, which lie at the heart of the TRESPASS process, effectively support change propagation across model instances. As shown in the figure, TRESPASS has applied the model transformation techniques to enable risk analysis (by the means of quantitative analysis on attack trees and other domain specific attack models) starting from the socio-technical model. This model transformation scheme enables efficient synchronization of the analysis models with the socio-technical model. This means that any modification in the socio-technical model will be automatically

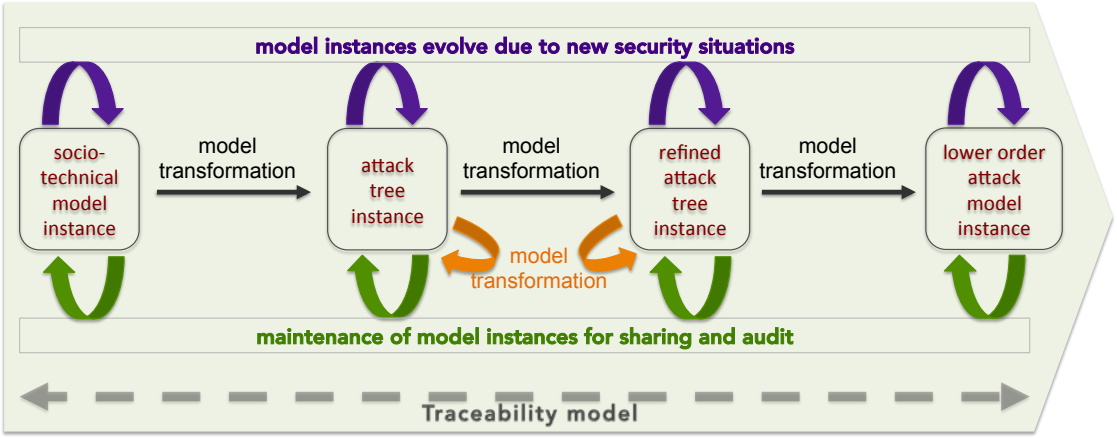

Figure 8.1: Conceptualization of the main model maintenance techniques in context of the TRE$_S$PASS model-based process.

propagated towards the analysis models. The TRE$_S$PASS traceability model supports traceability among different models. One of its main features, which is implemented in the ANM, is the model element identifiers that allow traceability from the attack model elements (e.g., a node in an attack tree) back to the related socio-technical model elements. This is one of the key enablers for the "what-if" analysis and selection of appropriate security policies (The TRE$_S$PASS Project, D3.4.2, 2016).

For model maintenance in dynamic situations, besides the means to introduce rapid updates in model instances and in the modeling formalisms themselves (discussed in Chapters 5 and 6), we have also evaluated the potential of dynamic risk assessment. This application has not been scaled to the whole TRE$_S$PASS process, but the results from the EpStan case study evaluation are very encouraging.

Other important model maintenance means implemented by TRE$_S$PASS but not featured in Figure 8.1 are the following:

- The information system that stores model instance versions in a lightweight git-like system. Versioning is an enabler for audit, as the auditor can analyze the history of model instances and data elements creation/usage.
- Extensibility of the socio-technical modelling language, the attack pattern library and other modelling paradigms. The means for extensibility ensure that new security scenarios that introduce previously unknown concepts can be handled by the TRE$_S$PASS process, and can be, for example, analyzed using the same tool set with minor modifications.
- Versioning and maintenance support for other types of TRE$_S$PASS models, such as ArchiMate and e3value.

Why are these best practices? This deliverable reports on the techniques designed in TRE$_S$PASS for model maintenance. These techniques were selected as best practices due to the following considerations:

- State-of-art for model-based processes recommends the use of model transformation and traceability links (as reported in Chapters 2-4). Therefore, these techniques were selected as the main drivers for model maintenance. Individual research results, such as the attack tree meta-model, transformation from socio-technical models into attack trees, transformations from attack trees into lower-level attack models, the traceability model, etc., have been published as scientific papers or have been successfully defended as Master theses results (and as such they were peer-reviewed by security and domain experts). Therefore, we can conclude that the choice of these techniques is accepted by the security community as the best practice.
- Evaluation and validation activities performed in TRE$_S$PASS have demonstrated that the model maintenance means designed by the project are appropriate. Notice that the project has not performed dedicated validation for all model maintenance practices, but all of them were evaluated as a part of the whole TRE$_S$PASS process validation on each case study (see the case studies deliverables (The TRE$_S$PASS Project, D7.2.2, 2016; The TRE$_S$PASS Project, D7.3.2, 2016; The TRE$_S$PASS Project, D7.4.2, 2016)). At the same time, many elements of the model maintenance techniques portfolio were validated in dedicated use cases and evaluated in interviews with practitioners (for example, the traceability model, dynamic risk assessment, extensibility and dynamic features of the socio-technical modeling language, compositional analysis of attack models, etc.). This allows us to conclude that the industry community of the project and the relevant domain experts (some of them not part of the project) have evaluated the designed model maintenance means and have not found deficiencies (The TRE$_S$PASS Project, D7.2.2, 2016; The TRE$_S$PASS Project, D7.3.2, 2016; The TRE$_S$PASS Project, D7.4.2, 2016; Syauta, 2016).
- We have learned a lot from peer EU research projects. For example, the lessons learned from the SHIELDS project reported in this deliverable and in (The TRE$_S$PASS Project, D5.3.1, 2013), have shown that the TRE$_S$PASS process needs to be decentralized, and will be better adopted by the community if there is no central dependency, but the analysts can select tools and techniques from the TRE$_S$PASS socio-technical risk assessment ecosystem according to their own needs. We have consequently adopted the same approach for model maintenance practices – while it has been designed with two main goals in mind, it is decentralized and individual tools and techniques for model maintenance can be selected by the analysts according to their needs. The lessons and ideas learned from the SecureChange project were a motivation for us to perform the traceability study. Moreover, we have used this project's results when designing techniques for model maintenance in dynamic situations. Thus, we have built our work on the European integrated research results.

8.3 Guidelines on selecting maintenance techniques

In this section we would like to share our experiences and lessons learned in model maintenance, and to provide advice for security practitioners and researchers who are developing their own model-based security process, and who wish to learn from TRE$_S$PASS model maintenance practices.

- *Start from the needs for maintenance.* We have been able to save a lot of time and effort by starting from identifying the requirements and needs for model maintenance that arise from the state of security practice. Our considerations about this state of practice are outlined in Chapter 1. These considerations were cross-checked with state-of-art in model-driven engineering and with experiences of peer European research projects. Starting from the two global goals for maintenance, we have then identified how these goals map into the needs of our case studies and how do they relate to the socio-technical security risk assessment process driven by TRE$_S$PASS.
- *Experiment with different maintenance techniques and tools in the early stages until the right one is found*. Once the requirements and needs have been identified, one can start experiments with different technologies to try to find the best fit. For example, we had originally planned to have a relational database for storing the data and model templates (the information system), but we found that a text-based system reliant on git is more practical for our case (see (The TRE$_S$PASS Project, D2.4.1, 2016) for more details).
- *Large processes need to be decentralized.* This advice is essentially a re-iteration from the legacy of the SHIELDS project. The TRE$_S$PASS tools ecosystem is large yet in essence decentralized, with many different tools and approaches interacting with each other. The analysts can adopt the parts according to their clients' needs. There is a much smaller risk of this ecosystem failing to be adopted in comparison with a single, vast, whole-piece framework. Similarly, in our opinion, it is not practical to design a general methodology for model maintenance for a large model-based process, but there is a need for a portfolio of techniques that can be selectively adopted by the analysts and practitioners based on their own preferences.
- *Provide maintenance means for each modelling language, but also ensure their integration and synchronization.* Following from the previous advice, if there is no monolithic framework but an ecosystem of interconnected tools, there should be means to maintain all individual model instances, and techniques to synchronize different types of modelling paradigms. In this way, the model maintenance techniques developed will ensure that the desired requirements for model maintenance are satisfied, while if the analysts opt to work with only certain elements of of the process, they will still be able to maintain the models and ensure their analyzability in the long run.

9 Conclusions

In this deliverable we have documented our approach for maintaining the TRE$_S$PASS models. Managing changing security requirements and addressing new security situations immediately in our analysis process is very important for acceptance of the TRE$_S$PASS results. Therefore, in this deliverable we reported our multi-faceted findings on this topic.

We have presented a summary of the relevant lessons learned from European research projects that we have used as the starting point for the TRE$_S$PASS model maintenance practices. We have reported on the traceability links maintenance study executed in the context of TRE$_S$PASS and the model transformation approaches used in TRE$_S$PASS to facilitate the maintenance tasks. We also reported on how the TRE$_S$PASS process elements handle dynamicity, including dynamic aspects of the TRE$_S$PASS socio-technical model, the evolving nature of virtualized environments in the cloud case study and the dynamic elements in the analysis models. All these techniques deal with changes as a top priority. Thus, by adding these features to the TRE$_S$PASS process we ensured smooth handling of dynamic security scenarios.

We have also reported on the maintenance in the presence of change of individual models used in the TRE$_S$PASS process. We have identified how the TRE$_S$PASS process deals with each important type of change and how to effectively propagate changes across the analysis. Furthermore, we have discussed dynamic risk assessment as a way to deal with changes in the environment. Last but not least, we have provided a high-level overview of the best practices developed and have shared advice on developing model maintenance means for such a complex security process as TRE$_S$PASS.

References

Ammar, M., Benaissa, M., & Chabchoub, H. (2015, May). Traceability management system: Literature review and proposal of a system integrating risk management for hazardous products transportation. In *Advanced logistics and transport (icalt), 2015 4th international conference on* (p. 229-234). doi: 10.1109/ICAdLT.2015.7136631

Aung, M. M., & Chang, Y. S. (2014). Traceability in a food supply chain: Safety and quality perspectives. *Food Control*, *39*, 172 - 184. Retrieved from http://www.sciencedirect.com/science/article/pii/S0956713513005811 doi: http://dx.doi.org/10.1016/j.foodcont.2013.11.007

Band, I., et al. (2015). *Modeling enterprise risk management and security with the ArchiMate® language.* The Open Group White Paper.

Bergomi, F., Paul, S., Solhaug, B., & Vignon-Davillier, R. (2013, Sept). Beyond traceability: Compared approaches to consistent security risk assessments. In *Availability, reliability and security (ares), 2013 eighth international conference on* (p. 814-820). doi: 10.1109/ARES.2013.109

Bleikertz, S., Vogel, C., & Groß, T. (2014, December). Cloud radar: Near real-time detection of security failures in dynamic virtualized infrastructures. In *Annual computer security applications conference, acsac 2014.* New York, NY, USA: ACM.

Boudali, H., Crouzen, P., & Stoelinga, M. (2010). A rigorous, compositional, and extensible framework for dynamic fault tree analysis. *Dependable and Secure Computing, IEEE Transactions on*, *7*(2), 128–143.

Clayton, R. (2005, November). *Anonymity and traceability in cyberspace* (Tech. Rep. No. UCAM-CL-TR-653). University of Cambridge, Computer Laboratory. Retrieved from http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-653.pdf

David, N., David, A., Hansen, R. R., Larsen, K. G., Legay, A., Olesen, M. C., & Probst, C. W. (2015). Modelling social-technical attacks with timed automata. In *Proceedings of the 7th acm ccs international workshop on managing insider security threats* (pp. 21–28).

ENISA. (2012). *ENISA Threat Landscape. Responding to the Evolving Threat Environment.*

European Commission. (2015). *Reform of the data protection legal framework in the EU, http://ec.europa.eu/justice/data-protection/reform/index_en.htm.*

Felderer, M., Katt, B., Kalb, P., Jürjens, J., Ochoa, M., Paci, F., ... Breu, R. (2014). Evolution of security engineering artifacts: A state of the art survey. *Int. J. of Secure Software Engineering*, *5*.

Felici, M., Meduri, V., Solhaug, B., & Tedeschi, A. (2011). Evolutionary risk analysis: expert judgement. In *Computer safety, reliability, and security* (pp. 99–112). Springer.

Gadyatskaya, O. (2015). How to generate security cameras: Towards defence generation for socio-technical systems. In *Proc. of gramsec* (Vol. 9390). Springer.

Gadyatskaya, O., Hansen, R. R., Larsen, K. G., Legay, A., Olesen, M. C., & Poulsen, D. B. (2016). Modelling attack-defense trees using timed automata. In *International conference on formal modeling and analysis of timed systems* (pp. 35–50).

Gadyatskaya, O., Harpes, C., Mauw, S., Muller, C., & Muller, S. (2016). Bridging two worlds: Reconciling practical risk assessment methodologies with theory of attack trees. In *Proc. of gramsec* (Vol. 9987). Springer.

Georgia Institute of Technology. (2014). *Emerging cyber threats report 2014,* https://www.gtisc.gatech.edu/pdf/Threats_Report_2014.pdf.

Ghamarian, A. H., de Mol, M., Rensink, A., Zambon, E., & Zimakova, M. (2012). Modelling and analysis using GROOVE. *Int. J. on Soft. Tools for Technology Transfer*, *14*, 15-40.

Gordijn, J., Akkermans, H., & Van Vliet, H. (2000). Business modelling is not process modelling. In *Conceptual modeling for e-business and the web, ecomo 2000* (Vol. 1921). Springer.

Huistra, D. (2016). *Automated generation of attack trees by unfolding graph transformation systems.* (Master thesis. University of Twente)

IEEE. (1991, Jan). IEEE standard computer dictionary: A compilation of IEEE standard computer glossaries. *IEEE Std 610*, 1-217. doi: 10.1109/IEEESTD.1991.106963

IEEE Guide for Software Requirements Specifications. (1984, Feb). *IEEE Std 830-1984*, 1-26. doi: 10.1109/IEEESTD.1984.119205

International Organization for Standardization. (1994). *Iso 8402: 1994: Quality management and quality assurance-vocabulary*. International Organization for Standardization.

Ionita, D., Wieringa, R., Wolos, L., Gordijn, J., & Pieters, W. (2015). Using value models for business risk analysis in e-service networks. In J. Ralyte, S. Espana, & O. Pastor (Eds.), *The practice of enterprise modeling* (Vol. 235, p. 239-253). Springer International Publishing. doi: 10.1007/978-3-319-25897-3_16

Ivanova, M. G., Probst, C. W., Hansen, R. R., & Kammuller, F. (2015). Transforming graphical system models to graphical attack models. In *Proc. of gramsec* (Vol. 9390). Springer.

Jhawar, R., Lounis, K., & Mauw, S. (2016). A stochastic framework for quantitative analysis of attack-defense trees. In *International workshop on security and trust management* (pp. 138–153).

Katta, V., & Stalhane, T. (2010). A conceptual model of traceability for safety systems. *CSDM-Poster Presentation*, 1–12.

Kordy, B., Mauw, S., Radomirović, S., & Schweitzer, P. (2011). Foundations of Attack–Defense Trees. In P. Degano, S. Etalle, & J. D. Guttman (Eds.), *FAST* (Vol. 6561, pp. 80–95). Springer.

Kordy, B., Mauw, S., Radomirović, S., & Schweitzer, P. (2012). Attack–Defense Trees. *Journal of Logic and Computation*, 1-33. (available online http://logcom.oxfordjournals.org/content/early/2012/06/21/logcom.exs029.short?rss=1) doi: 10.1093/logcom/exs029

Lund, M. S., Solhaug, B., & Stølen, K. (2010). Evolution in relation to risk and trust management. *IEEE Computer*, *43*(5), 49–55.

Lund, M. S., Solhaug, B., & Stølen, K. (2011). Risk analysis of changing and evolving systems using coras. In A. Aldini & R. Gorrieri (Eds.), *Foundations of se-*

curity analysis and design vi (Vol. 6858, p. 231-274). Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-642-23082-0_9 doi: 10.1007/978-3-642-23082-0_9

Massacci, F. (2012). *SecureChange project. Final publishable summary.* *http://www.securechange.eu/content/year-3-summary*.

Mens, T., & Van Gorp, P. (2006). A taxonomy of model transformation. *Electronic Notes in Theor. Comp. Science*, *152*, 125–142.

MITRE. (2016a). *Common attack patter enumeration and classification,* *https://capec.mitre.org/*.

MITRE. (2016b). *Common vulnerabilities and exposures,* *http://cve.mitre.org/*.

Moe, T. (1998). Perspectives on traceability in food manufacture. *Trends in Food Science & Technology*, *9*(5), 211 - 214. Retrieved from http://www.sciencedirect.com/science/article/pii/S0924224498000375 doi: http://dx.doi.org/10.1016/S0924-2244(98)00037-5

Olsen, G. K., & Oldevik, J. (2007). Scenarios of traceability in model to text transformations. In D. H. Akehurst, R. Vogel, & R. F. Paige (Eds.), *Model driven architecture-foundations and applications* (Vol. 4530, p. 144-156). Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-540-72901-3_11 doi: 10.1007/978-3-540-72901-3_11

Paci, F., Massacci, F., Bouquet, F., & Debricon, S. (2012). Managing evolution by orchestrating requirements and testing engineering process. In *Proc. of icst* (p. 834 - 841). IEEE.

Paige, R. F., Olsen, G. K., Kolovos, D. S., Zschaler, S., & Power, C. (2008). Building model-driven engineering traceability classifications. In *Ecmda traceability workshop (ecmda-tw).*

Paul, S., & Delande, O. (2011). Integrability of design modelling solution. *SecureChange FP7 project deliverable D4.4b*, *4*, 4b.

Pieters, W., Barendse, J., Ford, M., Heath, C., & Probst, C. (2016). The navigation metaphor in security economics. *IEEE Security & Privacy*.

Probst, C. W., Willemson, J., & Pieters, W. (2016). Graphical Models for Security: Second International Workshop, GraMSec 2015, Verona, Italy, July 13, 2015, Revised Selected Papers. In S. Mauw, B. Kordy, & S. Jajodia (Eds.), (pp. 1–17). Cham: Springer International Publishing. Retrieved from http://dx.doi.org/10.1007/978-3-319-29968-6_1 doi: 10.1007/978-3-319-29968-6_1

Refsdal, A., Solhaug, B., & Stølen, K. (2015). Risk management. In *Cyber-risk management* (pp. 9–24). Springer.

Refsdal, A., Solhaug, B., & Stølen, K. (2015). Security risk analysis of system changes exemplified within the oil and gas domain. *International Journal on Software Tools for Technology Transfer*, *17*(3), 251-266. Retrieved from http://dx.doi.org/10.1007/s10009-014-0351-0 doi: 10.1007/s10009-014-0351-0

Regattieri, A., Gamberi, M., & Manzini, R. (2007). Traceability of food products: General framework and experimental evidence. *Journal of Food Engineering*, *81*(2), 347 - 356. Retrieved from http://www.sciencedirect.com/science/article/pii/S0260877406006893 doi: http://dx.doi.org/10.1016/j.jfoodeng.2006.10.032

Sauff, H., & Gollmann, D. (2012). *Risk analysis for container transport.* Invited Talk at MetriSec 2012, http://metrisec2012.cs.nku.edu/MetriSecV2.doc.

Seehusen, F., & Solhaug, B. (2012). Tool-supported risk modeling and analysis of evolving critical infrastructures. In G. Quirchmayr, J. Basl, I. You, L. Xu, & E. Weippl (Eds.), *Multidisciplinary research and practice for information systems* (Vol. 7465, p. 562-577). Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-642-32498-7_43 doi: 10.1007/978-3-642-32498-7_43

Seibel, A. (2011, March). From software traceability to global model management and back again. In *Software maintenance and reengineering (csmr), 2011 15th european conference on* (p. 381-384). doi: 10.1109/CSMR.2011.58

Sendall, S., & Kozaczynski, W. (2003). Model transformation – the heart and soul of model-driven software development. *IEEE Software*.

Solhaug, B., & Seehusen, F. (2014). Model-driven risk analysis of evolving critical infrastructures. *Journal of Ambient Intelligence and Humanized Computing*, *5*(2), 187-204. Retrieved from http://dx.doi.org/10.1007/s12652-013-0179-6 doi: 10.1007/s12652-013-0179-6

Syauta, K. J. (2016). *Traceability in cyber risk assessment* (Unpublished master's thesis). TU Delft.

Symantec. (2015). *Internet security threat report, Volume 20, April 2015* http://www.symantec.com/security_response/publications/threatreport.jsp.

The MoVE Project. (2013). *MoVE - Model Versioning and Evolution.* http://move.q-e.at/. ([Online; accessed December 6, 2015])

The Open Group. (2016). *ArchiMate® 3.0 specification*. Van Haren Publishing.

The TRE$_S$PASS Project, D1.3.2. (2015). *Extensibility of socio-technical security models.* (Deliverable D1.3.2)

The TRE$_S$PASS Project, D1.3.3. (2015). *Dynamic features of socio-technical security models.* (Deliverable D1.3.3)

The TRE$_S$PASS Project, D1.3.4. (2016). *TRE$_S$PASS socio-technical security model and specification languages.* (Deliverable D1.3.4)

The TRE$_S$PASS Project, D2.2.2. (2015). *Data extraction from virtualized infrastructures.* (Deliverable D2.2.2)

The TRE$_S$PASS Project, D2.2.3. (2016). *TRE$_S$PASS technical data extraction tools.* (Deliverable D2.2.3)

The TRE$_S$PASS Project, D2.3.1. (2014). *Social data and policy extraction prototype.* (Deliverable D2.3.1)

The TRE$_S$PASS Project, D2.3.2. (2015). *TRE$_S$PASS social data and policy extraction techniques.* (Deliverable D2.3.2)

The TRE$_S$PASS Project, D2.4.1. (2016). *TRE$_S$PASS information system.* (Deliverable D2.4.1)

The TRE$_S$PASS Project, D3.4.1. (2014). *Attack generation from socio-technical security models.* (Deliverable D3.4.1)

The TRE$_S$PASS Project, D3.4.2. (2016). *Methods for attack generation, preventive measures, and ranking.* (Deliverable D3.4.2)

The TRE$_S$PASS Project, D3.5.1. (2016). *Dynamics of stochastic models.* (Deliverable D3.5.1)

The TRE$_S$PASS Project, D4.2.2. (2016). *Methods for visualization of information security risks.* (Deliverable D4.2.2)

The TRESPASS Project, D4.3.2. (2016). *Visualisations to simplify complex information.* (Deliverable D4.3.2)
The TRESPASS Project, D4.3.3. (2016). *Visualizations of socio-technical dimensions of information-security risks.* (Deliverable D4.3.3)
The TRESPASS Project, D5.3.1. (2013). *Abstraction levels for model sharing.* (Deliverable D5.3.1)
The TRESPASS Project, D5.3.2. (2015). *Best practices for model creation and sharing.* (Deliverable D5.3.2)
The TRESPASS Project, D5.4.2. (2016). *The integrated TRESPASS process.* (Deliverable D5.4.2)
The TRESPASS Project, D6.2.2. (2015). *Final refinement of functional requirements.* (Deliverable D6.2.2)
The TRESPASS Project, D6.4.4. (2016). *The integrated TRESPASS tools.* (Deliverable D6.4.4)
The TRESPASS Project, D7.2.2. (2016). *Final report case study a.* (Deliverable D7.2.2)
The TRESPASS Project, D7.3.2. (2016). *Final report case study b.* (Deliverable D7.3.2)
The TRESPASS Project, D7.4.2. (2016). *Final report case study c.* (Deliverable D7.4.2)
van Dorp, K.-J. (2002). Tracking and tracing: a structure for development and contemporary practices. *Logistics Information Management*, *15*(1), 24–33.
Wieringa, R. (1995). An introduction to requirements traceability. In *Faculty of mathematics and computer science, vrije universiteit, tech. rep. ir-389.*