Traceability in Cyber Risk Assessment

A design science approach

Kevin Joseph Syauta

Technische Universiteit Delft

TU Delft

Traceability in Cyber Risk Assessment

A design science approach

by

Kevin Joseph Syauta

to obtain the degree of Master of Science in Management of Technology
at the Delft University of Technology
to be defended publicly on Monday August 29, 2016 at 16:00 PM.

Student number: 4414063
Project timeline: March 2016 – July 2016
Thesis committee:

Prof. dr. ir. M.F.W.H.A. Janssen,	***Chair***, Section Information and Communication Technology
dr. ir. W. Pieters,	***First supervisor***, Section Safety and Security Science
dr. J. Hulstijn,	***Second supervisor***, Department of Management, Tilburg University

Section Information and Communication Technology
Department of Engineering Systems and Services
Faculty of Technology, Policy, and Management

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

predict
prioritise
prevent
TRESPASS

Preface

The thesis project is meant to be a learning experience ☺

The above quote was taken from an e-mail and was most probably not purposed to be inspirational, but it has resonated with me through the course of authoring this thesis. What lies in front of you is a fully-grown culmination of seeds of curiosity and hard work, organically nurtured for a few hundred days with countless cups of coffee as its fertiliser.

I wish to express my heartfelt gratitude to Wolter, the author of the quote who has very patiently guided me in completing this thesis and who has gratuitously allowed me to be involved in his research project. Your expertise and passion for cyber security have inspired me to deliver my best for this thesis, with hopes that someday and somehow it can be of benefit to a larger group than its intended audience. The academic and moral support you have given will never be forgotten. I would also like to thank Joris, who has been very thorough in giving his feedback, for his confidence in me and making me aware of things that needed more of my attention. My gratitude also goes to Prof. Marijn, whose warmth, support, and sharp eyes have helped shape this thesis the way it is written now. Of all the teachers that have taught me, I would like to especially thank Prof. Jan van den Berg who has instilled in me a love for cyber security; a field I have not even heard of when I first set foot in Delft. The researchers in TRESPASS also deserve a special mention in this thesis. I thank them for having set aside a few hours from their hectic schedule and donating their priceless ideas through verbal and non-verbal conversations. Especially to Chris, Axel, and Dan, thank you for having spent time to respond to my many e-mails and being very supportive of my ideas. I fully hope this work can contribute to TRESPASS beyond the way it is intended to.

More people deserve recognition for their role in making this thesis complete. First and foremost I wish to convey my gratitude to the Justus & Louise van Effen Foundation who has provided full financial support for me to be able to follow this Master's programme here. My family back home in Indonesia, thank you for the trust, prayers, endless support, and for being there (and also here) when I need you. Relatives and family members in the Netherlands, all the way from Amstelveen to Almelo: thank you for every single thing you have blessed me with. To all the close friends I have made in The Netherlands: thank you for all the good times; I'll treasure every moment we have spent together! To the close friends in Indonesia: as they say nowadays, see you at the top.

This thesis is dedicated to the memory of my late grandfathers. I wish you a good time in reading this work and I hope you learn something new out of it.

Kevin Joseph Syauta
Delft, July 2016

...there is nothing new under the sun.

(Ecclesiastes 1:9)

Summary

This Master thesis is written in partial fulfilment of requirements to graduate from the Management of Technology (MoT) Master programme. It elaborates a design science approach to incorporate traceability in the field of cyber risk assessment. It is done within the context of the European research project Technology-supported Risk Estimation by Predictive Assessment of Socio-technical Security (TRESPASS), whose goal is to support European organisations in improving their security posture in particular with respect to the socio-technical risks they are facing.

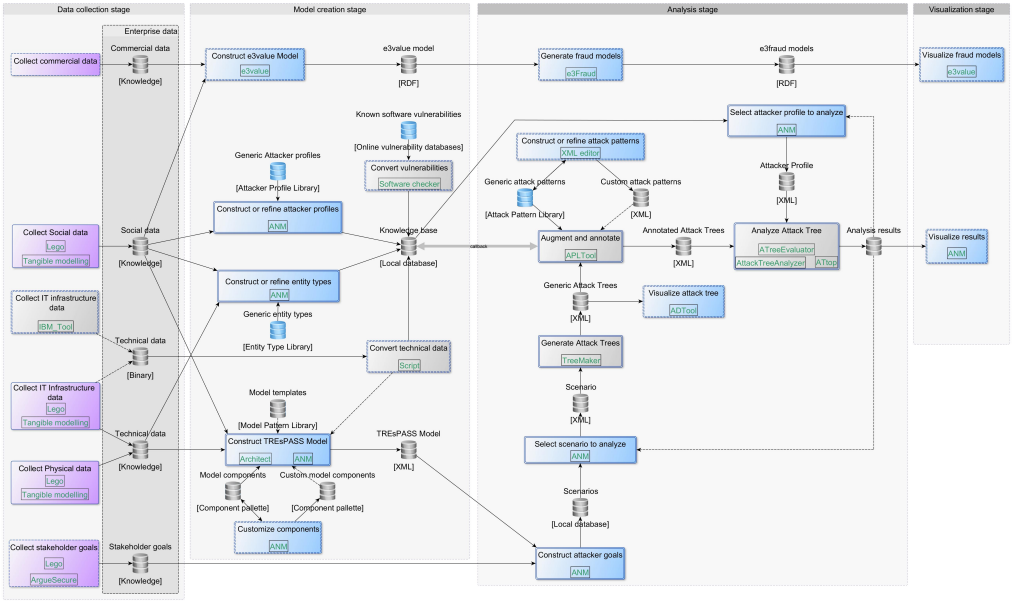
Technology has become such an integral part of our daily life. Radical technological innovations have taken the world by storm, giving us new phenomena such as the big data regime and the uberisation of businesses. Technologies such as cloud computing solutions, Internet of Things (IoT), temporary social media, and Apple Pay are becoming hot items demanded by individuals and organisations. The World Economic Forum believes that the world is standing in the front door of the so-called Fourth Industrial Revolution or Industry 4.0, when cyber-physical systems will prevail and it will be harder to draw a line between the physical and digital realms. This *cyber dependency* global trend produces risks. We call these *cyber risks*; those that if not managed properly could potentially lead to events with unwanted consequences in both digital and physical realms.

The Allianz Risk Barometer indicates cyber risk as one of the top three global business risks in 2016. However, data breaches and other cyber attacks that often colour the headlines could have been prevented had high-tech organisations devoted more attention to addressing cyber risks. A possible trajectory to achieve this is by conducting a thorough risk assessment as part of managing risks emanating from technologies used in the organisation. However, a lot of processes are involved in the risk assessment process - collecting internal and external data for the assessment (*context establishment*), building a model of the organisation and identifying the risks the organisation is facing (*risk identification*), running the analysis (*risk analysis*), making sense of the analysis results (*risk evaluation*), and deciding what to do to address the risks (*risk treatment*). A cyber risk assessment is difficult for a few reasons. Firstly, the socio-technical nature of cyberspace implies that the organisation needs to consider physical, digital, and organisational realms in conducting the risk assessment and involve a chock-full of inputs from all three domains. Secondly, due to this wide scope of risk assessment, coming to a complete understanding of the results would be difficult. Let us assume that the organisation has now understood the implications of the results and has formulated a list of risk treatment decisions to be taken. The organisation eventually has to be able to return to the results of the tedious risk assessment to prove that the decisions were made objectively based on the results and that people had agreed on these decisions. Lastly there is the problem of dealing with changes. Can they account for the impact of these changes to their risk posture without having to run a complete risk assessment process from scratch? Issues like these can be addressed by incorporating *traceability* into the risk assessment process, which can intuitively be translated as the capability to empower the users to understand the dependencies and connectivity of process, data, and the model involved in the risk assessment exercise. However, this may not be an exhaustive definition of traceability, let alone accurate. This lack of understanding - which then leads to a scarcity of traceability support in risk assessment models - needs to be rectified. This research attempts to first fill the gap of understanding this concept by providing a definition and specification of traceability in the field of cyber risk assessment.

Having explained these problems, we formulate our main research question (RQ) as follows:

How can the notion of traceability be incorporated in cyber risk assessment?

As mentioned in the preamble, this research is performed within the context of TRESPASS project. TRESPASS is a risk assessment model that supports decision makers in performing risk assessment in socio-technical systems. The TRESPASS tools achieve this by facilitating a complete risk assessment activity, starting from collecting the data needed to perform the assessment (*data collection*), building a model to represent the socio-technical system of interest (*model creation*), analysing the risk posture of the system (*analysis*), and finally visualising the analysis results to assist decision makers prioritise their decisions (*visualisation*). An overview of these activities is shown in the next page. However, TRESPASS also suffers from the generic problem with risk assessment models not having a support for traceability. Hence, we also aim to generate a support for traceability for TRESPASS in the form of a *trace model*. The trace model is a framework that is able to provide traceability in TRESPASS by building links spanning from the data used in the risk assessment process to the results of the risk assessment activity. In this



Source: TREsPASS project

thesis, different models of cloud computing environment are addressed as cases of a socio-technical system viewed from a corporate perspective.

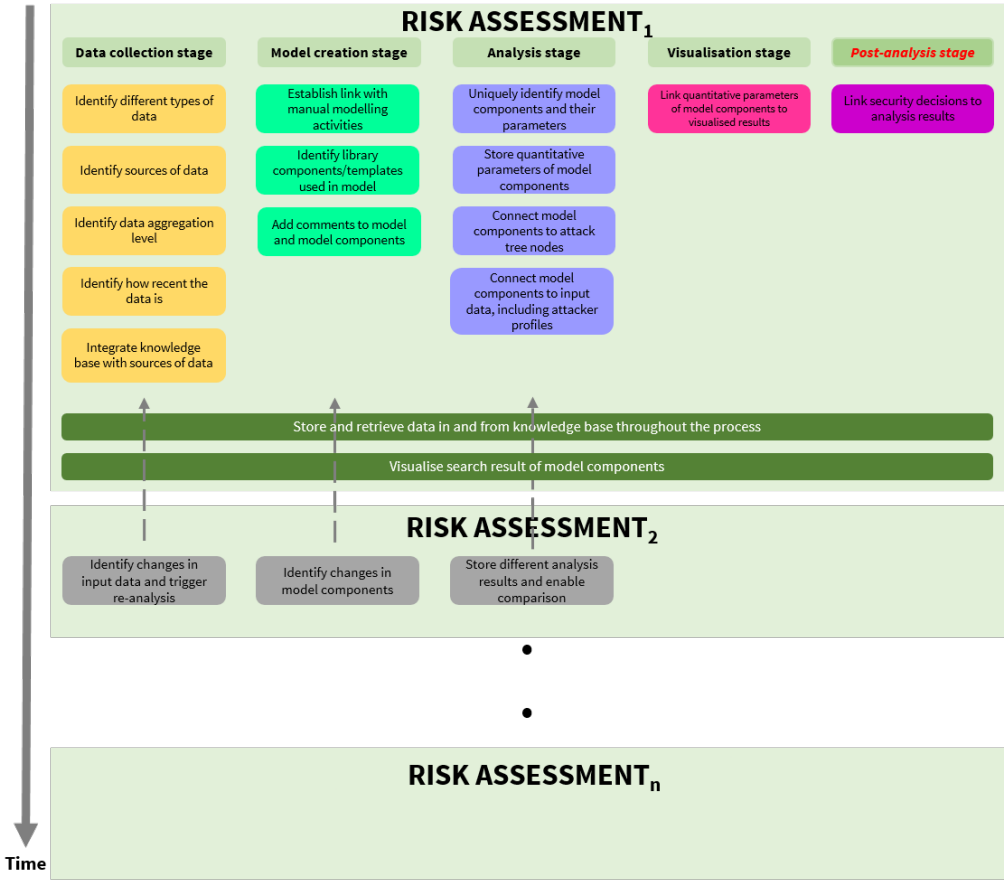
To answer the RQ, we first investigated the notion of traceability in cyber risk assessment by referring to literature in other domains such as food industry and requirements engineering and by interviewing experts within TREsPASS to induce their opinions. This mixture of input led us to discover that 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.*

Departing from this definition, 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. The final features of the trace model are summarised in an overarching figure shown below.



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. However, implementation of the trace model was not considered in the research due to the fact that the project is ending in a few months.

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. We also invite researchers to explore the notion of traceability as a value that might be inherently present in risk assessment, by making use of value sensitive design (VSD) approach. 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.

Contents

List of Figures

List of Tables

1 Introduction

In this foremost chapter, the reader can find an overview of the underlying problem statement. We first sketch a bird's eye view of the problem prior to explaining the practical relevance of the research. Also presented here are the objectives of the research, as well as the corresponding research questions and sub research questions. We briefly explain the chosen research approaches before outlining the expected contributions of this thesis. Finally a short section that presents the overall structure of the thesis wraps this chapter up.

1.1. Problem statement

1.1.1. Scientific problem

People, things and organisations around the world are becoming more digitally interconnected. The World Economic Forum refers to this trend as *cyber dependency* [223]. Consequently, cyber attacks and data fraud or theft have landed in the list global risks in 2016 in terms of likelihood. Cyber risks have to be soundly managed to help the world navigate towards what is often dubbed the Fourth Industrial Revolution (or Industry 4.0 according to Allianz Risk Barometer), where boundaries between physical and digital realms are becoming more blurry [162]. The Internet of Things (IoT) will play a huge role in increasing our cyber dependency, with an estimation of 20 to 30 billion connected devices worldwide in 2020[1]. Failure to manage cyber risks will lead to a significant spike in cybercrime. It is estimated that cybercrime has cost the world USD 445 billion in 2014 [121] and will cost the world USD 2 trillion in 2019[2]. Organisations need to be more cognisant on the importance of addressing cyber risks. Allianz Risk Barometer places risk of cyber incidents among the top three global business risks in 2016 [9]. Verizon's 2016 edition of Data Breach Investigation Report (DBIR)[3] identifies *miscellaneous errors* ("*any unintentional action or mistake...*") as the top cause of data breaches happening across industries, followed by incidents caused by insiders and misuse of privilege, and physical theft and loss [212]. 2015 was dubbed the Year of Collateral Damage by Hewlett-Packard, as attacks happened in organisations who thought they were bulletproof [182]. The U.S. Office of Personnel Management breach in June 2015 gave away 21 million accounts and the Ashley Madison hack in July 2015 exposed almost 100 GB of data [212]. Organisations fail to implement even the most basic security controls, because if they did, four out of five cyber attacks would not have happened [181]. The 2016 DBIR also mentions that 93% of all breaches identified happened within only minutes or even less, while it took 83% of the victimised organisation weeks or more to discover the breach.

Cyber dependency coupled with criminals with strong financial motivation [212] shape today's cyber risk landscape [8]. Emerging technological trends, such as BYOD (Bring Your Own Device) and cloud computing can potentially bring benefits to high-tech organisations, such as savings on corporate devices and increased productivity[4]. Cloud computing in particular has gained significant popularity among organisations within the last decade because of the benefits it offers compared to traditional information technology (IT) solutions, such as on-demand self service and rapid elasticity [124]. On the other hand, the use of these technologies could create potential perils to organisations. A disruption at a cloud provider's physical infrastructure (e.g. power outage or data breach) could interrupt the

[1] `http://www.mckinsey.com/industries/high-tech/our-insights/the-internet-of-things-sizing-up-the-opportunity`
[2] `http://www.juniperresearch.com/press/press-releases/cybercrime-cost-businesses-over-2trillion`
[3] Verizon distinguishes *breaches* from *incidents*. An incident is defined as a security event that compromises the CIA properties of an information asset (cf. Section 2.1). A breach is understood as an incident that results in involuntary disclosure of data.
[4] `http://searchmobilecomputing.techtarget.com/essentialguide/How-to-weigh-BYOD-benefits-and-risks`

business activities and incur significant loss [8]. Gartner researchers indicate a list of risks to be considered when utilising cloud computing, such as risks on data privacy and data segregation [76]. The fact that these risks exist should not discourage organisations from embracing cloud computing. If appropriately managed and assessed, these risks can be transformed into opportunities.

As a way to cope with risks emanating from the use of technology, organisations can practice risk management using standards such as those covered by ISO 31000. Three main activities are involved in risk management process: *communication and consultation* with stakeholders in the organisation to interact and share information; *risk assessment*; and *monitoring and review* of current status and continuous check-up of the risk management process [147]. According to ISO, risk assessment is the "overall process of risk identification, risk analysis, and risk evaluation" [93]. This definition exhibits three different activities: *risk identification*, which seeks to discover and describe risks in the organisation; *risk analysis* which explores the sources of risks, their impact, and the likelihood that the impact actually materialises; and *risk evaluation*, which consists of evaluating the level of risk analysed in the previous step against the organisation's risk criteria. Sometimes the activities of *context establishment* (setting the risk assessment objectives as well as risk criteria) and *risk treatment* (looking for options to deal with the evaluated risks) are included in the risk assessment process [147].

When the risk assessment activity is specifically focused to deal with cyber risks, we call it cyber risk assessment. This focus might bring the false illusion that cyber risk assessment is easier to be done, as the scope is supposedly narrowed down to only assessing the risks emanating from technical infrastructure. As with any other risk assessment activity, a cyber risk assessment exercise should never segregate the organisation's technical infrastructure from its physical infrastructure and its organisational context. Cyber risk assessment is a complex activity in itself. A proper assessment requires the organisation to carefully document their information assets and the supporting technical infrastructure, including its vulnerabilities. They also need to comprehend their network infrastructure, make sense of the organisational context (e.g. mapping out access control policies), and build a *risk model* from all this information [147]. We have not even talked about understanding adversaries with their different strategies and resources. To understand this complexity, let us examine an organisation utilising cloud computing technologies as an example.

Cyber risk assessment for technologies such as cloud computing is difficult for several reasons. Firstly, a cloud computing environment along with the people interacting with it constitute a socio-technical system [133]. The socio-technical nature of a cloud computing environment demands a risk assessment model that can cover the complex technical infrastructure, physical infrastructure and human factors [24]. An assessment of solely the technical infrastructure, however meticulous, will produce an incomplete risk posture of the cloud computing and brings additional risk of missing out attack vectors that are introduced from the physical realm of the organisation. Secondly, cloud computing is essentially a form of outsourcing activity done by organisations, which introduces new entities to be considered during the risk assessment. Other tenants that are residing in a common cloud facility orchestrate a systemic risk [8]. Interruption experienced by one customer could produce a ripple effect that affects other customers. Cloud customers also experience risks due to diminished control over their data. The obligations to comply with rules and regulations will raise various new concerns (Where will the data reside? Who will take care of it? Can confidentiality be preserved?). Lastly, the dynamic and flexible nature of an organisation may cause a one-time off risk assessment to be invalid. Cloud computing can easily facilitate changes [120]. Continuous risk assessment needs to be done to reflect both internal changes (e.g. business-justified changes such as system upgrades or changes in organisation policies) and external changes (e.g. changes in laws and regulations). In this thesis, we address cloud computing as a case to focus on.

When all is said and done, and the risk manager has conducted a tedious cyber risk assessment activity which generates massive results with hundreds of variables, a new problem spawns: how does the organisation come to a complete understanding of the results? Why is their CEO marked as very susceptible to fall prey to a phone-based social engineering attack? Why does the result indicate that the most possible person to jeopardise the virtual machines is actually the janitor? Results that sometimes challenge common sense require full comprehension, and *traceability* does have to potential to aid this. Traceability here might be translated as the capability to empower the users to understand the dependencies and connectivity of process, data, and the model involved in the risk assessment exercise. By explicitly showing how the data and the components of the organisation are used in the risk assessment process, the organisation may arrive at a solid understanding of the risk assessment results and can formulate informed decisions thereafter.

That is, however, not the entire story. After the relevant people have made the security-related decisions (risk treatment options), how do they ensure that such decisions are defensible? In case anything goes wrong, or when the organisation is unfortunately hit by an attack it did not anticipate in the first place, the decision makers who determined which controls to invest in and apply may need to revisit why such decisions were made. This shows

another instance where traceability could assist. Had the decision makers clearly stated these decisions and how they relate to the analysis results (e.g. *"Upgrade all virtual machines as they showed high level of vulnerability and may contribute to a successful attack by competitors."* and *"Put off awareness trainings until the next fiscal year due to lack of budget and low likelihood of fruitful social engineering attacks."*), traceability could provide clear tracing back to these rationales.

Finally, our risk assessment difficulties increase when we factor time into the equation. External and internal changes may impact the overall risk posture of the organisation. Traceability would have an acute role in for instance, understanding how organisational and infrastructural changes would affect the level of cyber risk that an organisation faces. Again, the socio-technical nature of an IT-driven organisation adds insult to the injury. The aggregated effects of technical and social changes are elusive and difficult to quantify. Here, traceability might help decision makers discover the root causes of a certain level of cyber risk that an organisation faces back to the events of changes that have materialised in the past.

These examples are only a subset of many potential benefits offered by establishing robust traceability in the cyber risk assessment process. Taking the above aspects into consideration, how can organisations ensure that the risk assessment models they use can comprehensively address the complex nature of socio-technical system such as a cloud computing environment? Why is there a risk of insider and privilege misuse? Why do we consider the cloud administrator as a potential threat to the organisation? How come the organisation is vulnerable to a denial of service (DoS) attack? The dynamic nature of cloud computing also makes it relevant to reflect and record relevant changes such that the risk analysis result remains valid. An example of such changes is when a new vulnerability is discovered in the access management solution. What new risks are introduced to the organisation? Are there possible new attack vectors? Is the organisation's overall risk posture affected by this discovery? Which attack scenarios and which elements of the system model are affecting the change in the level of risk? A traceability-supported cyber risk assessment model should be able to help organisations answer these questions. However, we see two problems here: firstly, the concept of traceability itself is hardly understood in cyber risk assessment. Defining and specifying traceability in this space remains an outstanding task. Secondly, due to this lack of understanding, a scarcity of traceability support exists in cyber risk assessment models in general. Most of current established risk assessment methodologies or tools do not consider traceability as a part of its standard offer. Our research aims to first define and specify the notion of traceability in cyber risk assessment and afterwards provide the support for traceability in a risk assessment model. Traceability will help high-tech organisation conduct a more robust risk assessment as part of its technology management processes.

1.1.2. Practical relevance

To cope with the ever-changing cyber risk landscape, cyber risk assessment models need to be able to assist organisations in predicting, prioritising, and preventing attacks. Moreover, the risk model needs to be able to visualise the results so that decision makers can take actions in a timely manner. Current established risk models fall short of either one of these requirements. Some models can provide a very complete risk picture but do not assist decision makers to make decisions rapidly. The TRESPASS (*Technology-supported Risk Estimation by Predictive Assessment of Socio-technical Security*) project aims to develop a model with such capabilities. It supports decision makers by building the so-called *attack navigator* that is capable of *assessing* possible attack paths, *calculating* the associated risks and *visualising* possible attack routes in socio-technical systems, in a manner very akin to a map [144]. This Europe-wide project is intended to "reduce security incidents in Europe and allow organisations and their customers to make informed decisions about security investments"[198] which in turn will help enforce *trust and security* as a key pillar of the Digital Single Market Strategy [56]. The project is expected to be finished by October 2016. Within the context of an organisation, information security risk management has become an indispensable part of risk management activities [87]. The role of a Chief Information Security Officer (CISO) or an Information Risk Manager becomes increasingly important in safeguarding an organisation's information assets. Such information-intensive organisations are the intended audience of the TRESPASS tools. Table 1.1 summarises the different contexts in which TRESPASS can be used in risk management in different types of organisations, loosely derived from [195].

Table 1.1: Context of uses of TRESPASS (Source: [195])

User	Functional goals/needs	Context
Chief Information Security Officer (CISO)	Analyse possible attacks happening to the organisation based on TRESPASS results and decide on appropriate security investments	Large organisations
Auditor	Assess whether an organisation complies with a certain security threshold, defined in a standard/referential	Large organisations
Risk manager	Evaluate the level of cyber risk exposed to the organisation based on TRESPASS results	Large organisations
Small and medium enterprise (SME) employee	Identify potential security weaknesses in product/service offerings to embrace security-by-design or to gain a quick scan of its security posture	SMEs

The Use Cases indicated in TRESPASS outline several possible uses of the TRESPASS model, for example to decide investments in information security in a large company, to identify potential weaknesses in product or service in an innovative enterprise, or to evaluate the potential impact of a newly proposed product/service on the security of the combined product portfolio in an organisation [195]. These Use Cases indicate potential usage of TRESPASS in a (dynamic) socio-technical system. As is the case with most risk models, TRESPASS model currently has neither a tool nor a model/framework to support traceability despite the fact that the project is approaching its maturity phase. We have acknowledged that this is a generic problem in cyber risk assessment models. In this research we aim to provide a support for traceability in TRESPASS after coming to an understanding of what the idea of traceability means in cyber risk assessment.

1.2. Research objective

This practice-oriented research's objective is twofold. Firstly, it tries to contribute to the cyber risk assessment body of knowledge by formulating a formal definition of traceability in that domain. This is done through literature study and garnering expert opinion.

Secondly, it seeks to contribute towards the development of TRESPASS model by generating support for traceability in the form of a *trace model*. A trace model is a framework that is able to provide traceability in TRESPASS by building links spanning from the data used in the risk assessment process to the results of the risk assessment activity. The integration diagram in TRESPASS shown in Figure 1.1 helps to communicate the overall TRESPASS process. There are essentially four different stages in the TRESPASS process: *data collection stage*, *model creation stage*, *analysis stage*, and *visualisation stage*. During the data collection stage, various types of data that are used in the risk assessment process are collected. A model of the target of analysis (e.g. the organisation to be assessed) is constructed during the model creation stage, in the form of the so-called TRESPASS model. The analysis stage involves assessment of the target of analysis using information provided in the previous stages. This process is done by means of generating attack scenarios, constructing an attack tree based on the scenarios, annotating it with relevant quantitative parameters, augmenting the attack steps in the tree and finally analysing the attack tree based on different criteria using different analysis tools. Finally results of the analysis are visualised during the visualisation stage in the form of an attack navigator. In this thesis we do not take into account the top part of the map (the e3 value fraud case study). Further explanation on the integration diagram is given in Appendix B.

The trace model is built with a design science approach using case studies in TRESPASS on cloud computing in a high-tech organisation. It is intended to explicitly track and link any changes in the system that affect the risk calculation of the TRESPASS model, but also enables backwards tracing: The risk analysis results can be linked to the attack trees. The attack trees can be further traced back to the corresponding attack scenarios, and even further back to the TRESPASS model. For example, the chance of a successful social engineering attack with the goal of stealing employee data is given as 0.57. This number is generated from a list of attack scenarios annotated with probabilities (*Annotated Attack Trees*), for instance one that involves the HR manager clicking on a spear phishing email. This specific attack scenario is linked with elements of the TRESPASS model, including the HR manager as an *Actor* and her laptop as a *Device Location*[5]. An even further step would be to trace how the TRESPASS model came into being from the modelling process and look into data used in the calculation and how they were retrieved. It should be noted however that this research will not consider the integration of the trace model into the overall TRESPASS tools. This research goes to the length of only devising its specifications such that a framework is formed but will not produce a tool support for it. In total, there are two example cases in this research: The first case tackles a *private*

[5]See Appendix A for explanation of concepts used in TRESPASS.

cloud computing environment while the second example depicts an *outsourced* cloud computing environment. Both cases are derived from the TRESPASS project library and publications and reflect a cloud computing environment in a high-tech organisation. Using two different contexts of cloud computing allow us to evaluate whether the trace model functions well in different contexts. Ultimately, this research delivers a validated trace model for TRESPASS that can be applied for different models of cloud computing.

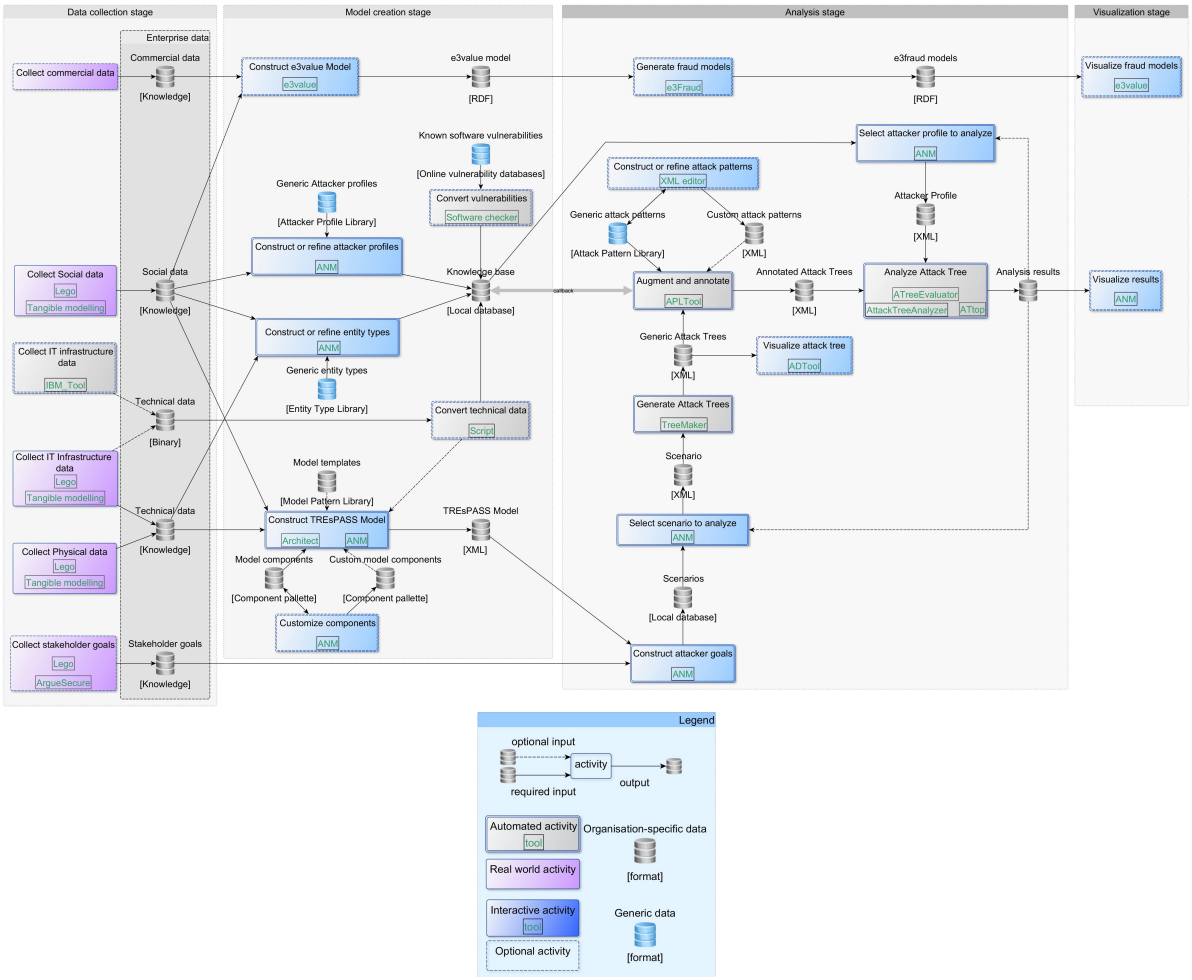


Figure 1.1: TRESPASS integration diagram

Source: [195]

This research is done internally within the TRESPASS project through collaboration with different partners. This research is part of WP5 (Work Package 5) which focuses on process integration, including extending the TRESPASS approach as it stands now. Results of this research will be disseminated as part of a deliverable within the TRESPASS project on best practices for model maintenance.

1.3. Research question + sub questions

In conjunction with the aforementioned scientific and practical problems that exist as well as our research objective, we formulate the following central research question (RQ) of this thesis as follows:

RQ: **How can traceability be incorporated in cyber risk assessment?**

To be able to answer this research question, we first have to be able to define the notion of traceability in the domain of cyber risk assessment and its requirements. This brings us to the first sub-research question, namely:

SRQ1: **How can the notion of traceability be defined and specified in the domain of cyber risk assessment?**

The answer to SRQ1 contributes towards existing knowledge base on traceability, which is currently dominated by literatures from supply chain and food industry. The question is formulated in an explanatory nature since there currently exists little to no works specifically discussing traceability in the domain of cyber risk assessment. A formal definition of traceability in cyber risk assessment is produced by drawing references from different domains and by consulting experts that are involved in TRESPASS . With this definition and specification of traceability in cyber risk assessment, we begin to explore and apply this notion in TRESPASS by building a support framework of traceability named the trace model. In building this support, we utilise a case study from TRESPASS on a private cloud computing solution in a high-tech organisation. This brings us to the second SRQ which is:

SRQ2: **What are the requirements to provide a traceability support for TRESPASS in the form of a trace model applied in a cloud computing environment?**

The answer to SRQ1 is our building block in answering SRQ2. When the elements of traceability have been clearly delineated, we can define functional requirements of the trace model. After the trace model is built, we test it with a different case to ensure that the trace model can adapt with different models of cloud computing. This finally brings us to the final SRQ as follows:

SRQ3: **How can the trace model be verified and tested to see whether it fulfils its requirements and is applicable to similar cloud environments?**

Verification is understood as the process to see whether the trace model satisfies its requirements by drawing expert opinions, and testing is understood as seeing whether the trace model works in a different context of cloud computing by conducting a single-case mechanism experiment on an outsourced cloud computing infrastructure. The whole activity is referred to as validation process. The term *validation* is used to preserve consistency with the steps involved in design science methodology according to Wieringa [221]. By answering all three SRQs, we arrive at an answer to the main RQ.

1.4. Research approaches

In this section, the chosen research approaches are explained. Trade-offs and justifications for the research approaches of choice are presented, followed by detailed articulation of the approaches. Further explanation is given in Chapter 3.

1.4.1. Generic trade-offs

The central RQ that is posed in this research is of *exploratory* nature. It seeks to embed a concept (traceability) that has currently has little to no relation with the subject of interest in this research (cyber risk assessment model). To answer this question, in-depth understanding of key concepts need to be built, which explains why the SRQs are also prescribed in *exploratory* nature. According to Verschuren and Dooreward [214], there are three key decisions that need to be made when selecting research approaches. These key decisions are:

1. Breadth versus depth
2. Qualitative versus quantitative research
3. Empirical versus desk research

The central RQ of this research implies the need for an in-depth understanding. Therefore, we decide to go for *depth* instead of *breadth* in this research. The research approaches of choice reflect this decision. The second and third key decisions are made by taking into consideration the overall research goals in the TRESPASS project. There are three kinds of research goals in the TRESPASS project: *design* (designing artefacts), *analysis* (analysing a conceptual artefact), and *empirical research* (studying the behaviour of objects or subjects) [222]. Our research fits into the first goal of the TRESPASS project. Using design science methodology, we attempt to design an artefact (the trace model) and validate it . Therefore, *qualitative* research is preferred as opposed to *quantitative* research. For the third key decision, we choose to conduct both empirical research and desk research. Further explanation on this decision is given in the following section.

1.4.2. Justifications of research approaches

We have explicitly stated the key decisions pertaining to the research approaches of this research. The various goals of this research calls for a *multi-method* approach. These research approaches differ according to the SRQ that we intend to answer. An overview of the relation between SRQs and the selected research approaches is shown in Table 1.2.

Table 1.2: Relation between SRQs and selected research approaches

No.	SRQ	Research approach	Reference(s)/Methods
1	SRQ1	Desk research + Expert opinion	Literature analysis [218] + Semi-structured interviews analysis [31]
2	SRQ2	Design science	[213] [221]
3	SRQ3	Expert opinion + single-case mechanism experiment	[221]

Desk research is needed to answer SRQ1. The process of constructing a concept of traceability that can be applied in the domain of cyber risk assessment is done by referring to existing works of traceability in different fields, hence desk research as the method of choice. We also elicit expert opinions by conducting semi-structured interviews with people involved in the TRESPASS project to gain more insights.

SRQ2 concerns a *design problem*. It seeks to revamp the TRESPASS model as it exists now, and it calls for a solution which in turn is result of a design. Design science is the suitable research approach to answer this SRQ because it enables us to design and evaluate a novel artefact (in this case, the trace model), and delivers a how-to knowledge. We do not resort to grounded theory approach to explore this SRQ because it is more suitable for a *theory-oriented* research, while this research is more *practice-oriented*. To execute the design science approach, we refer to the methodology explained in [213] and [221] .

The third SRQ shall be answered through verification sessions with experts and a single-case mechanism experiment, collectively referred to as the validation step. The goal of this step is twofold: to check whether the trace model satisfies its requirements and whether it functions properly in a different context in which it was designed. Expert opinion is once again chosen not only to sieve bad design, but also to predict how the trace model will behave [221]. The single-case mechanism experiment is intended to help observe how the trace model responses to different inputs.

1.5. Expected contributions

We envision two important contributions from our research. Firstly, we expect to enrich the risk assessment body of knowledge by formulating a definition of traceability along with the components that constitute it for the domain of information risk assessment based on literature research and soliciting expert opinions. Secondly, we intend to contribute to the TRESPASS project with a trace model that provides support for traceability as we define it in cyber risk assessment, built based on design science approach using examples of different cloud computing architectures. However, it is not our intention that the trace model is only applicable to TRESPASS - contributions of the trace model for risk assessment practice in general will be inferred by generalisations of its features where applicable.

1.6. Thesis structure

This thesis is divided into eight different chapters. An overview of how the chapters are connected with one another is shown in Figure 1.2. The contents of each chapter is given as follows:

1. **Introduction**

   This chapter consists of problem statements, research objectives, research questions and sub research questions, explanation of the research approaches used in the thesis, the expected contributions of this thesis, and the structure of the thesis.

2. **State of the art**

   This chapter introduces the reader to principal definitions used in the research and gives an overview state of the art researches within the related domains. Current knowledge gaps are also presented in this chapter.

3. **Methodology**

   This chapter unravels the process of doing systematic literature analysis and design science methodology in general and how we intend to approach the research using these approaches. All three phases of the design science methodology are thoroughly explained here.

4. **Defining and specifying traceability in cyber risk assessment**

This chapter contains the answer to SRQ1, that is the definition and specification of traceability within the domain of risk assessment in general based on literature survey from multiple disciplines and semi-structured interviews with experts.

5. **Problem investigation**

   This chapter concerns the first phase of the design science methodology, containing the description of the stakeholders, their expectation of the research and their goals, as well as the phenomena that is investigated and how they contribute to stakeholders' goals.

6. **Treatment design**

   This chapter concerns the second phase of the design science methodology, containing the requirements and context assumptions for the treatment, currently available treatments, and the design process itself.

7. **Treatment validation**

   This chapter contains the results of verification and testing activities of the trace model, including explanations on whether the trace model fulfils its requirements and whether it is applicable to other cases on cloud computing.

8. **Conclusions and discussion**

   This chapter contains the main findings of the thesis and the answers to the (sub) research questions. A discussion on the implication of the findings and limitations of the thesis is also given in this chapter. Lastly, directions and pointers for future research are also outlined in this chapter.

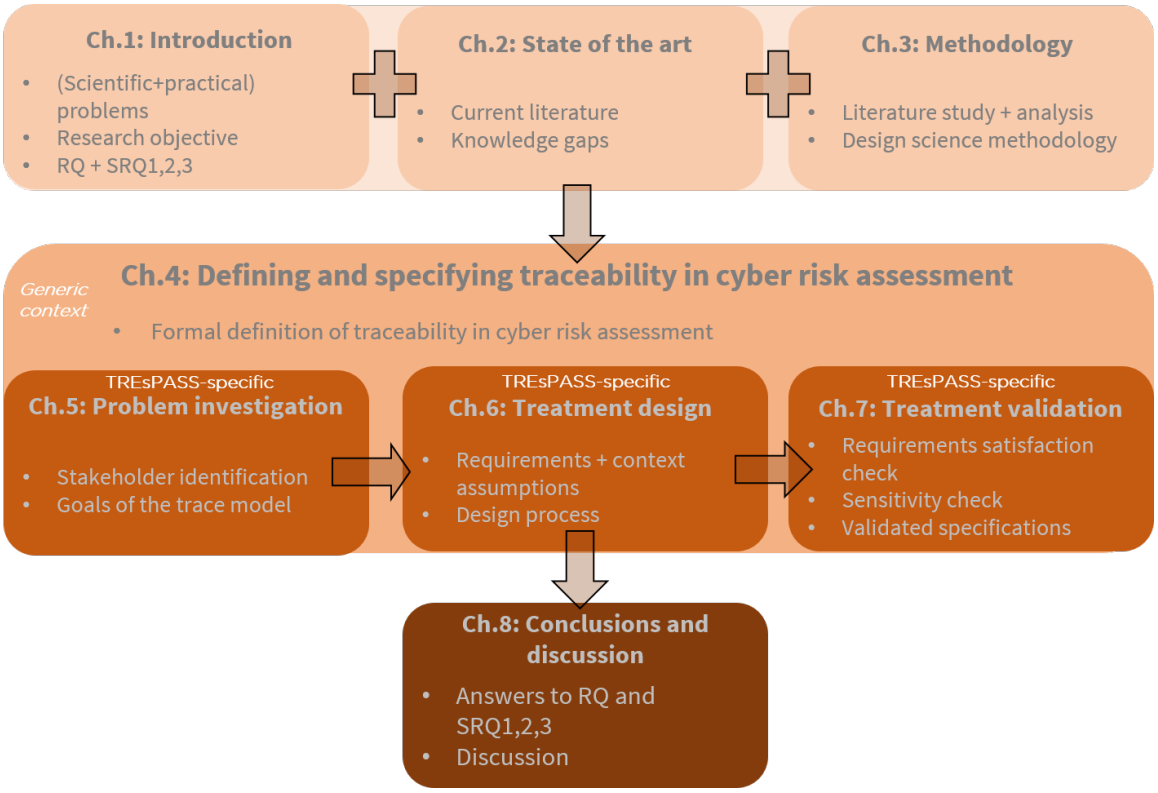


Figure 1.2: Overall logic of the thesis

2

State of the art

This chapter presents an overview of existing works that provide the baseline for this research, identifies existing knowledge gap, and outlines the contributions of this thesis. It opens with elaboration of key concepts that are relevant to cyber risk assessment. Following that is an extensive discussion around the overall process of the TRESPASS project. Subsequently, the notion of traceability in general and in several domains are explained. Key terminologies and principles of cloud computing are also elucidated to help readers understand the case studies used in this thesis. We end the chapter by stating the knowledge gaps we identify and the contributions of our research.

2.1. Cyber security, cyber risk, and cyber risk assessment

This section gives the reader a crash course on important concepts relating to cyber risk assessment. We depart from explaining what cyberspace is and move on to arguing that information security and cyber security are two different concepts. We then explain what cyber risk is and finally provide a discourse on what cyber risk assessment implies.

It is important to establish a common ground on what cyberspace means. We subscribe to the elaboration of cyberspace by van den Berg et al, who conceptualise that cyberspace is made up of three layers [202]. The layer located at the core is the *technical layer*, consisting of the Internet and its constituent technologies. On top of it is the *socio-technical layer*, where people conduct all sorts of Internet-enabled activities ranging from watching television series to radicalising teenagers with extreme beliefs. The outermost layer is the *governance layer*, where state and non-state actors produce policies, regulation, and standards to govern the former two layers. The conceptualisation is graphically shown in Figure 2.1. With this understanding, we advocate that cyberspace extends beyond merely the Internet or inter-connected networks. Referring to the glossary of National Institute of Standards and Technology (NIST), cyber security is understood as the ability to guard cyberspace from cyber attacks [107].

It is widely agreed that information is an asset (i.e. something of value) in the cyberspace. Information security is concerned with a triad of three notions: *confidentiality*, *integrity*, and *availability* (CIA) [62]. Confidentiality refers to preserving the security and privacy of information and preventing unwanted access, integrity means preserving the reliability of information and preventing unwanted manipulation, and availability boils down to preserving the ability of a system to work under usual conditions. Others have tried to expand these requirements by for example adding *authenticity* (proving you are who you claim to be), *non-repudiation* (not being able to deny an association to something), and *accountability* (responsibility towards information assurance) [202]. From the NIST glossary, information security concerns the protection of information systems [107]. Following our view on cyberspace, we see that information systems is only a part of cyberspace. Hence, information security is not equal to cyber security, but *part of* cyber security.

Pursuant to the definition of cyber security from NIST, cyberspace is prone to cyber attacks emanating from threats to the cyberspace. This implies that there are risks in the cyberspace. We refer to these risks as cyber risks. We agree with the view of Refsdal et al on cyber risk as risk caused by a cyber threat [147]. We refrain ourselves from decomposing risk in this chapter. Working definitions on risk and related concepts are laid out in Appendix A. As we use the term cyber risk, we also imply socio-technical risk due to the presence of the socio-technical layer in the cyberspace.

As with any other risks - financial risk or environmental risk - cyber risk has to be properly managed. The

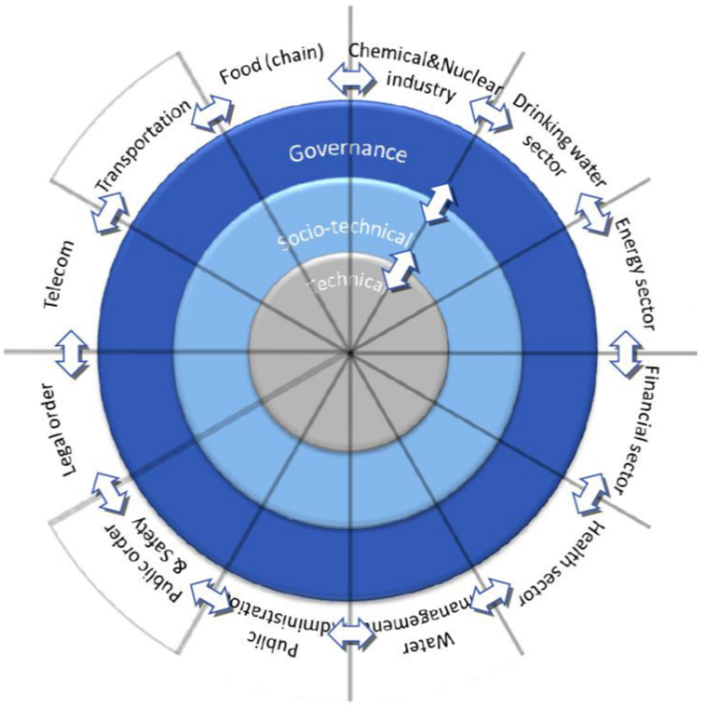


Figure 2.1: Conceptualisation of cyberspace

Source: [202]

overall process to manage cyber risk in an organisation or a system is called cyber risk management. We have mentioned in Chapter 1 that risk management consists of three activities: *communication and consultation*; *risk assessment*; and *monitoring and review*. The same activities can be applied to cyber risk management, but cyber risk management is special for two reasons [147]:

- The nature of cyberspace makes it possible that the stakeholders extend beyond the organisation; essentially it is possible that there are stakeholders anywhere in the world; and
- The adversarial nature of cyber risk exponentially increases the attack surface of the organisation; essentially it is possible that the organisation has adversaries anywhere in the world.

Within the broader context of cyber risk management, our focus is concentrated towards cyber risk assessment, which is a set of activities on understanding and documenting the organisation's cyber risk exposure [147]. Figure 2.2 shows that risk assessment should be conducted in an iterative manner. Variations exist, but a typical risk assessment activity consists of the following five steps [90] [147]:

1. **Context establishment**: Understanding the context, goals, and objectives of the risk assessment activity. Also done here are defining the scope of the risk assessment, formulating assumptions, and establishing the risk evaluation criteria.
2. **Risk identification**: Documenting the possible causes of risk, starting from the source of the threat to the incident [147].
3. **Risk analysis**: Estimating and determining the level of risks identified.
4. **Risk evaluation**: Comparing the level of cyber risks analysed to the risk evaluation criteria previously defined.
5. **Risk treatment**: Selecting the proper treatment for the identified risks. Usually there are four options [94]: *accept* the risks as part of 'business as usual', *avoid* the risks by eliminating the activity causing the risks, *mitigate* the risks by applying controls, and *transfer/share* the risks by for example going for cyber insurance or outsourcing the activity related to the risk.

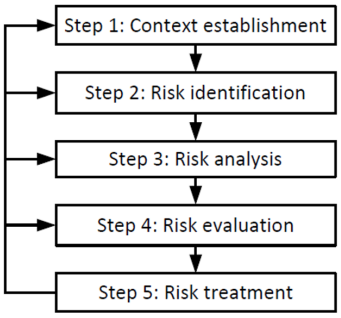


Figure 2.2: Risk assessment process

Source: [147]

As mentioned in Chapter 1, ISO does not consider the first and fifth step as part of risk assessment. This, however does not significantly affect our discussion.

It is not precisely known when cyber risk assessment originated, but one of the earliest information risk assessment methods, *CRAMM (CCTA Risk Analysis and Management Method)* came out at 1985 [1]. Since then more cyber risk assessment methodologies and tools have been launched over the past decades, for example *OCTAVE (Operationally Critical Threat, Asset, and Vulnerability Evaluation)* in 1999 [6],*CORAS* in the early 2000s [90], *FAIR (Factor Analysis of Information Risk)* in 2005 [100], and *TARA (Threat Agent Risk Assessment)* in 2009 [151]. There also exist various frameworks and standards on cyber risk management, such as ISO/IEC27005:2011 on information security risk management [94] and the NIST Guide for Conducting Risk Assessment [152]. Methodologies and tools for information security risk assessment are well-documented in [90]. Some of these methods undergo routine maintenance, but some have ceased to update.

Risk assessment methodologies can be classified according to several criteria. A popular manner to classify risk assessment methodologies is to base them on the risk scales they employ: *quantitative* methodologies use ratio and difference scales, whereas *qualitative* methodologies represent risk in nominal and ordinal scales [147]. Qualitative methodologies usually classify risks in categories from (Very) Low up to (Very) High. An example of a quantitative methodology is ISAMM (Information Security Assessment & Monitoring)[2]. OCTAVE exemplifies a qualitative methodology. There are also methods that combine both approaches, such as CORAS and FAIR [90]. Houmb classifies risk assessment methods and tools into three broad categories: *rule-based*, *risk-based*, and *judgment-based* [82]. Rule-based methods are characterised with extensive brainstorming sessions and regular meetings with stakeholders, and they are normally suitable to be adopted by standards. Risk-based methods target both known and unknown undesired events (risks) and focus on identifying and assessing the probability of these events and are suitable for complex organisations and critical systems. Finally, judgement-based methods rely on the subjectivity of experts' judgements.

Other possible categories to assign to these methodologies and tools include *probabilistic models*, such as Bayesian Attack Graphs (BAG) [143] that represent causal dependencies between network states; *adversarial models* such as adversarial risk analysis that combines statistics and game theoretic approach [15]; and *attacker models* such as TARA that places heavy emphasis on properties of attackers. The following section discusses TREsPASS as an example of a state of the art approach to cyber risk assessment. As we are about to learn, TREsPASS can be classified as a tool that combines both qualitative and quantitative approach.

2.2. The TREsPASS project

The TREsPASS project aims to assist humans in taking security decisions about large, complex infrastructures, by creating an *attack navigator* [144]. It helps decision makers by highlighting and visualising possible and most urgent attack opportunities and effective countermeasures against them [198]. TREsPASS suits the profile of a risk-based method. Compared to the previously established methods and tools in [186], TREsPASS comes with several features, namely [186]:

[1] As recorded by ENISA (European Union Agency for Network and Information Security) at `https://www.enisa.europa.eu/topics/threat-risk-management/risk-management/current-risk/risk-management-inventory/rm-ra-tools/t_cramm.html`

[2] `https://www.itrust.lu/wp-content/uploads/2007/09/publications_TTC_2007_abstract_risk_assessment_with_ISAMM.pdf`

- Integration with established standards and methodologies;
- Integration with databases of common vulnerabilities and threats;
- Support for automated attack generation attacks, allowing a proactive security assessment by predicting how changes to the model impact attack attributes; and
- Consideration of different profiles of attackers using different parameters (e.g. resources, goal, set of skills) to identify attacks and countermeasures on a system model, and to predict the likelihood of success and impact of the attack.

2.2.1. Components of the TRESPASS model

A socio-technical security model such as TRESPASS needs to be capable of abstracting the physical, digital, and social aspects of an organisation. As outlined in [188], there are several requirements in the TRESPASS model, namely:

- **Infrastructure**: TRESPASS model abstracts *physical*, *digital*, and *social* layers of an infrastructure.
- **Assets and containment**: Assets are represented in TRESPASS in various ways, such as *data* or *items*. Containment is enabled through location attributes, for example it is possible to represent a file as an asset that is contained in another asset (a Laptop).
- **Processes, Actions, and Behaviour**: In TRESPASS, processes are represented as metaphoric locations, which allows for representation of data storage in a process.
- **Policies**: TRESPASS policies represent a set of credentials that enable a set of actions.
- **Quantitative measures**: TRESPASS supports quantitative annotations (e.g. probability, cost of countermeasure, or impact of attack) for model elements.
- **Attacks, Vulnerabilities and Countermeasures**: TRESPASS represents vulnerabilities and countermeasures qualitatively.

The complete set of requirements for socio-technical security models is presented in D1.1.2 [188], while the functional requirements are given in D6.2.2 [195]. TRESPASS consists of typical components one would expect in a socio-technical security model, including [92]:

- **Actors**: persons or roles relevant to the model
- **Locations**: containers, including *physical*, *network*, and *device* locations.
- **Assets**: tangible *items* and *data* relevant to the model, for example a file, or an access credential.
- **Relationships**: including *position* and *possession* of actors, *containment* of network/device locations, and *connection* of locations.
- **Access Policies**: access enablers to locations given the right credentials (in the form of item or data assets).

A glossary of the core concepts used in TRESPASS is given in Appendix A.

The focus of WP5 lies in integrating the overall TRESPASS process, including the extensibility of the TRESPASS model. Extensibility as outlined in D1.3.2 encompasses three different areas: *internal embeddings*, *extensions to the language*, and *tool support* [189]. The document shall lay foundation for our research in providing support of traceability for TRESPASS, as it contains formal requirements on how these three types of extensions can apply to TRESPASS process. However, we first need to be able to define the concept of traceability, (specifically in the context of risk assessment) to be able to define how the trace model should look like. This notion is unravelled in Section 2.3.

2.2.2. The navigation metaphor in TRESPASS

One of the uniqueness of TRESPASS lies in its navigation metaphor. TRESPASS aids humans in studying socio-technical security of a complex system, which is instantiated in the *attack navigator map* [144]. The 'navigation' process is described as follows [141] [144]:

1. Initial modelling activities helps visualise the target of analysis in a *satellite view*, in which an aerial, high-level overview of the organisation is provided without revealing many bits and pieces;

2. The reality (organisation) is abstracted as a 'map' through what is termed a *system model*, which contains components such as *Actors, Assets, Locations, etc.* (cf. Section 2.2.1);

3. An *attacker goal* (stated as a policy violation) is used to explore plausible ways to navigate through the system model. Indeed, attacker properties are of paramount importance to devise possible navigation routes. An attacker's goals can follow from his/her profile, appended with assumptions on probability, resources (available time and cost of attack), or utility for the attacker; and

4. The attack navigator derives *feasible routes* to the specific attacker goal and their properties based on information on the attacker profile (skill and resources), represented as *attack trees* [141].

The attack navigator map helps an organisation explain and justify economic considerations regarding security decisions [141]. It provides answers to questions such as *What optimal strategies should be used to prevent a specific attack scenario?, How effective is this specific countermeasure when applied in our organisation?, How does it reduce the likelihood of success of a specific attack vector?* et cetera. In short, the navigation metaphor in TRESPASS can provide better understanding of a system's weak links and consideration for different attacker profiles with different strategies or utility functions, which in turn will support appropriate security investments [141].

The navigation metaphor has been adapted to offer a service model in TRESPASS, with steps outlining possible interaction between TRESPASS consultants and their clients [199]. More details are given in Chapter 6 (Section 6.3), where we use the service model description to outline the TRESPASS processes to aid the construction of the trace model using different attack scenarios.

2.2.3. TRESPASS cloud case study

Within TRESPASS, a number of case studies have been done with different socio-technical systems. These include Internet Protocol Television (IPTV), telecommunication systems, cloud computing, and Automated Teller Machines (ATM) [190]. As cloud computing is the domain application of this research, this section shall briefly introduce the reader to the cloud case study used in the TRESPASS project. There are different versions of cloud infrastructures used within TRESPASS. Nidd et al [133] made use of a simplified cloud environment to show how a tool-based risk assessment can be carried out. The restricted deliverable D7.2.1 elaborates the cloud case study in detail. An example of outsourced cloud infrastructure is used by Ionita et al to demonstrate an argumentation-based risk assessment approach [91]. Finally, a complete representation of a cloud infrastructure with its physical and organisational setting has been used in the Social Engineering Challenge 2015, a competition seeking a realistic social engineering attack scenario. This cloud setup (shown in Figure 2.3) is chosen to guide the development of the trace model because it sufficiently articulates the physical, technical, and social dimensions of a cloud computing environment viewed as a socio-technical system. Albeit simplified, necessary details are provided in each dimension to properly represent a realistic cloud computing environment. Granular details on the cloud model and the corresponding scenarios are given in Chapter 5, section 5.2. The outsourced cloud infrastructure model (shown in Figure 2.4) is used during the testing process of the trace model and is further explicated in Chapter 7. It is chosen for its architectural difference from the design case.

2.3. Traceability

Traceability as a notion is interpreted differently in various domains. In [11], these different definitions are outlined. For example, in computing, traceability has to do with products and development process [86]. In quality management, traceability is more related with recorded identification of an article or an activity [89]. In the domain of food industry, there are four different contexts in which the term traceability can be used [127]:

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

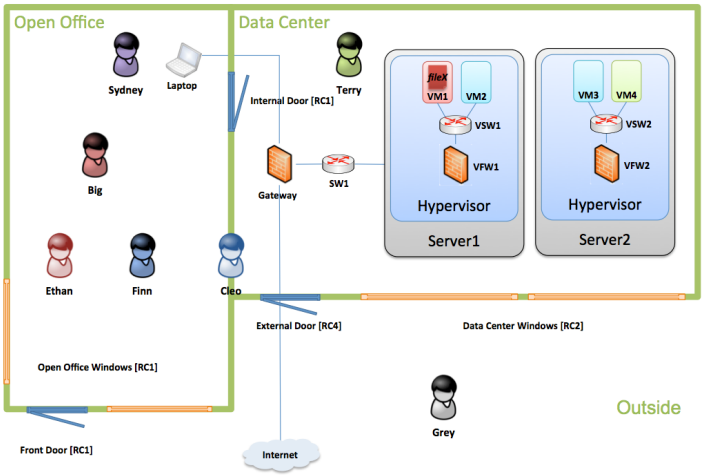


Figure 2.3: Cloud model used in the design process

Source: [187]

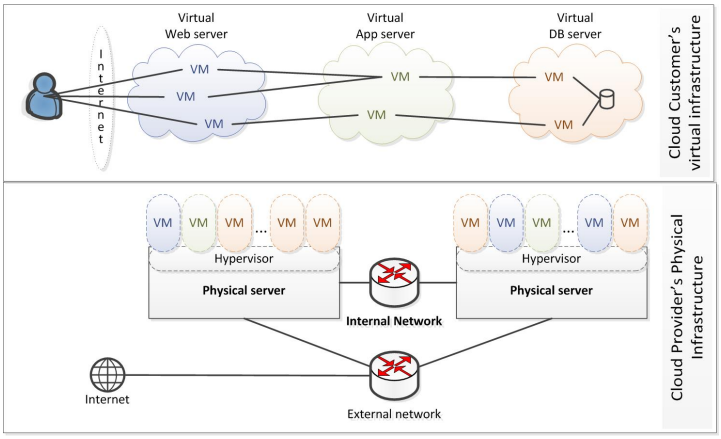


Figure 2.4: Cloud model used in the testing process

Source: [91]

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 2.3.1, 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.

2.3.1. Traceability in different domains

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 [11]. 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 [14]. 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 [149]. 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) [127]. 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 [127], which includes official control of foodstuffs and liability for defect products [209]. It is made possible through *item coding* and *information architecture* [209].

Traceability in software development has gone from the ability to trace requirements down to code (*requirements traceability*) [165] to "the capability for tracing artefacts along a set of chained operations..." (*software traceability*) [135]. 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." [1]. Traceability in model-driven engineering, for instance, can be established through *trace links* that connect different sources to their respective target elements [134]. 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 [220]. An example of integration of traceability into software development is the MoVE (Model Versioning and Evolution) project [183].

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 [127].

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 [40]. 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.

2.3.2. 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 [171]. In CORAS, this is also known as *evolutionary risk analysis* [57] [116]. According to Lund et al, there are three different perspectives with regards to changes in a system [117]. 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)[3]

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 [147] [164].

In TRESPASS project, the concept of change is captured in [190]. The document assesses the dynamic features of the TRESPASS 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 TRESPASS model to cope with such changes. The document also lists some dynamic features in the case studies. Concerning the cloud case study, the dynamic features are presented in Section 2.4.3. 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 TRESPASS 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 [117]. This is aligned with the TRESPASS case study on

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

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, 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 (see Section 2.4.2) 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 [101], 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 [164] [171] [148]. 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. This is shown in Figure 2.5. Although traceability is extensively discussed in CORAS, a formal definition for traceability is not specified.

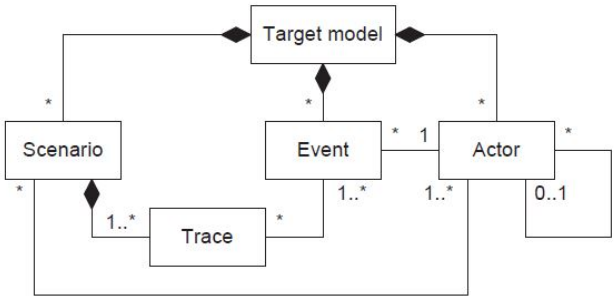


Figure 2.5: Trace model relationship in CORAS

Source: [171]

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 [138]. A comparison between CORAS and Rinforzando is presented in [21]. 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 [164]. 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. We end our discussion on traceability here, and in the succeeding section we present cloud computing as the field in which the research focuses on.

2.4. Virtualisation and the clouds

Up to now we have discussed cyber risk as the overarching topic of our research, TRESPASS as our object of study to be developed further, and traceability as the central notion to be unearthed in our research. In this section we will explore cloud computing as the application domain. We first explicate the concept of virtualisation, followed by important concepts in cloud, and finally risks and threats of cloud computing.

2.4.1. Virtualisation

Virtualisation is a term that is often confused with cloud computing. One has to bear in mind that virtualisation is *not* cloud computing *per se*, but it is an integral part of a cloud computing environment. In principle, virtualising means separating a resource for a service from the underlying physical delivery of that service [215]. NIST defines virtualisation as "simulation of the software and/or hardware upon which other software runs" [79]. Various types of virtualisation exist, such as *application virtualisation*, *operating system (OS) virtualisation*, and *full virtualisation* [79]. Two of the most important technologies underlying virtualisation are:

- A *virtual machine (VM)* is a software computer that emulates the functions of a physical computer, running with its own OS and applications;
- A *hypervisor* is a software that acts as a platform for VMs. Two types of hypervisor exists: Type 1 or *bare metal* hypervisor and Type 2 or *hosted hypervisor*. A bare metal hypervisor runs on a "blank" computer (no OS installed) and needs to work with the help of a *management software*, while a hosted hypervisor runs on a computer which already has a specific OS installed in it.

The hypervisor can host multiple VMs on a single physical server, which can run multiple types of OS. Virtualisation maximises a server's full capacity and hence driving higher efficiency. This is known as partitioning [215]. Despite its benefits, organisations should weigh in the risks that come with virtualisation [73]. These risks stretch beyond merely the technology (e.g. data protection risks) and include organisational processes, such as asset management and patch management risks. From a security point of view, virtualisation also poses threats to confidentiality, availability, integrity, authenticity, and non-repudation properties [201].

2.4.2. Cloud computing

According to NIST, cloud computing is defined as follows:

> *"Cloud Computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model promotes availability and is composed of five essential characteristics, three service models, and four deployment models."* [65]

These characteristics, service models, and deployment models as defined by NIST are briefly mentioned as follows:

1. **Essential Characteristics**
   - **On-demand self-service**
   - **Broad network access**
   - **Resource pooling**
   - **Rapid elasticity**
   - **Measured service**
2. **Service Models**
   - **Cloud Software as a Service (SaaS)**: Consumers can use provider's applications that run on a cloud infrastructure.
   - **Cloud Platform as a Service (PaaS)**: Consumers can deploy self-created or acquired applications using languages and tools supported by the cloud provider.
   - **Cloud Infrastructure as a Service (IaaS)**: Consumers are provided fundamental computing resources to deploy and run arbitrary software.
3. **Deployment Models**
   - **Private cloud**: The cloud infrastructure is operated solely for an organisation.
   - **Community cloud**: The cloud infrastructure is shared by several organisations.
   - **Public cloud**: The cloud infrastructure is made available to the general public or a large industry group.

- **Hybrid cloud**: The cloud infrastructure is made up of two or more clouds (private, community, or public).

NIST also presents a three-layered framework for a generalised cloud system, given in Figure 2.6 [80] as follows:

- **Service layer**: Where the cloud provider provisions each of the three service models (SaaS, PaaS, and IaaS).
- **Resource abstraction and control layer**: Where software elements are located, such as *hypervisors, virtual machines, virtual data storage* and other resource abstraction and management components.
- **Physical resource later**: Where physical computing resources are located, such as *computers, networks*, and other physical computing infrastructure elements.

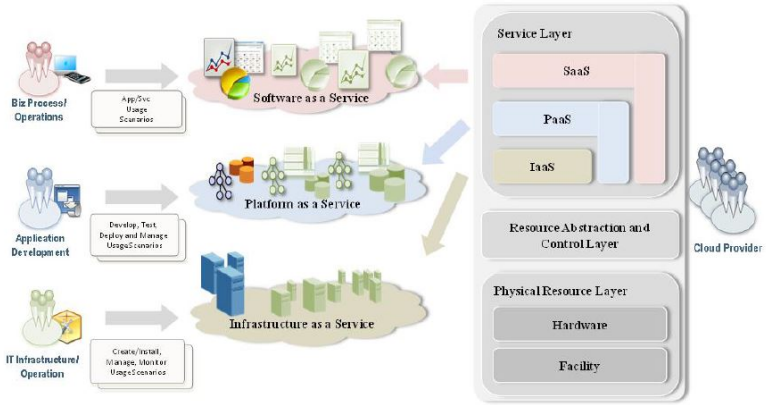


Figure 2.6: A three-layered framework for cloud

Source: [80]

2.4.3. Risks and attacks in a cloud environment

Due to the many benefits it brings [65], cloud computing has rapidly developed and become popular among businesses computing. Taking into account its flexibility, an organisation's cloud computing environment can undergo different changes, hence the term *dynamic cloud*. It "gives companies what they need, where they need it and how and when they want it." [120]. For example, a company decides to free up resources in its cloud by deleting a number of VMs, or it may want to develop three new softwares at a time in the cloud.

The dynamics of cloud brings forth challenges in terms of vulnerabilities and risk. Using terminologies from FAIR, [71] indicate four areas of cloud-specific vulnerabilities as follows: core-technology vulnerabilities, essential cloud characteristics vulnerabilities, defects in known security controls and prevalent vulnerabilities in state-of-the-art cloud offerings. Gartner mentions seven different aspects to evaluate when assessing the security risks of cloud computing, namely *privileged user access, compliance, data location, data segregation, availability, recovery, investigative support, viability*, and *support in reducing risk* [76]. [108] differentiate risks according to the perspectives of cloud provider (CP) and cloud customer (CC), shown in Table 2.1. Finally, [37] divides the risk according to the different layers of abstraction in cloud computing, summarised in Table 2.2.

Table 2.1: Cloud risks according to providers' and customers' perspective (Source: [108])

Aspect	Risks
Data security and privacy	Ensure availability of customer's data in cloud (CP)
	Risks related to data security and privacy (CP), (CC)
	Preventing unauthorised access to customer's data in the cloud (CP), (CC)
	Risks related to multi-tenancy (CP)
	Risks related to data deletion (CP)

Table 2.1 Cloud risks according to providers' and customers' perspective (continued)

Aspect	Risks
Technology	Lack of standardised technology in the cloud computing system (CC)
	Compatibility issue between cloud and IT systems in customer's organisation (CC)
Organisational	Risks related to resource planning, change management (CC)
	Risks related to security management (CC)
Physical security	The physical security of a cloud provider's data centres composed of servers, storage, and network devices (CP)
Compliance	Enforce regulatory obligations in a cloud environment (CP)
	Business continuity and disaster recovery (CP)

Table 2.2: Risks at different layers of cloud computing (Source: [37])

Layer	Aspect	Risks
Infrastructure (IaaS)	Network level	Connection pooling; Liaison attacker/eavesdropping; Interruption of nodes
	Host level	Buffer overflows; Physical theft of storage account information
	Application level	Brute force attacks
	Data communication and storage level	Data tampering
Platform (PaaS)	Code vulnerabilities	Disclosure of confidential data; File system of registry tampering
	Data storage and access	Open ports listening in insecure interfaces
	Auditing and logging	Upgrade of a privilege
Application (SaaS)	Secrecy/privacy	Dictionary attacks; Weak encryption
	Identity and access management	Cookie manipulation; Cookie replay attack; Information disclosure
	Authentication and authorisation	Cross-site Request Forgery (CSRF); Cross-site Scripting (XSS)
	Application level	Request flooding; Misconfiguration of service setting or application setting

A cloud environment is vulnerable to different types of attacks. As we shall see, the types of attacks mentioned in literature revolve around the *technical layer* of the cloud environment, leaving out possible attack scenarios stemming from the *physical layer* or *organisational layer*. These kinds of attacks is documented in [167] as follows:

1. **Common attacks**

   - **Distributed Denial of Service Attacks (DDoS)**: Exploitation of a number of VMs as internal bots to flood the victim's VM(s) with malicious requests.
   - **Keystroke timing attacks**: An attempt to measure the time between keystrokes when typing a password.
   - **Side-channel attacks**: A cross-VM attack (information leakage due to sharing of physical resources) through placement of malicious VM on the same physical machine as that of the target.

2. **Cloud-specific attacks**

   - **VM Denial of Service (VM DoS) attacks**: Exploitation of a vulnerability in the hypervisor to consume available resources of the physical machine on which the VM is running.
   - **Hypervisor attacks**: Penetration into guest VMs through the hypervisor by misuse of privileged access.
   - **Cloud malware injection attacks**: Injection of a malicious VM into the Cloud with different mischievous purposes.

- **VM image attacks**: Risk of disclosing sensitive information, running and distributing malicious images over the cloud network.
- **VM relocation attacks**: An adversary steals a victim's VMs and copies it to another location.
- **Resource-freeing attacks**: Exploitation of resource sharing among VMs to modify a victim VM's workload to release resources for the adversary's VM.
- **Fraudulent resource consumption attacks**: Exploitation of utility pricing model of the cloud through an attack similar to a DDoS attack.

Various attempts have been done to assess specific risks in cloud infrastructure. These works encompass a wide variety of proposed methods. For instance, an adjustable risk assessment method for cloud service providers and users is presented in [38]. Another risk assessment framework with special focus on service life cycle is proposed in [52]. A systematic literature review of risk assessment in cloud computing reveals that certain topics such as data security and privacy have been widely investigated, while others, e.g. physical and organisational security, have received less attention [108]. According to Creese et al [47], the current best practices in information security risk controls as defined by the ISO27001/27002 standards and the NIST Recommended Security Controls for Federal Information Systems and Organizations Special Publication 800–53 Revision 3 are inadequate for the cloud as the controls being put in place in these standards are falling behind the necessity for appropriate protection. These findings place forward the need for a novel socio-technical perspective in assessing risk in cloud computing.

Within the TRESPASS project, various works have been done to model risk in cloud infrastructures. For instance, Bleikertz et al propose a comprehensive model of cloud infrastructure with emphasis on security objectives of cloud customers and archetypes of attackers' characteristics [24]. The model is demonstrated using a set of known scenarios as well as counterfactuals. Bleikertz et al also put forth an automated security analysis system termed *Cloud Radar*, which presents near real-time capability to detects misconfigurations and security failures [24]. Nidd et al present a preliminary risk model on cloud infrastructure [133]. In this model not only complex technical infrastructures of cloud computing are considered, but also the human factors and the physical infrastructure. Within TRESPASS , the dynamic features of a cloud model have been evaluated in [190]. This includes the *physical infrastructure, virtual infrastructure, social organisational, social-technical,* and the *environmental status.*

2.5. Knowledge gap and research contributions

The result of the literature review reveals the following knowledge and practical gaps:

- Traceability in cyber risk assessment is yet to be defined and specified;
- TRESPASS needs to be extended to support traceability.

In sum, the expected contributions of this research are:

- A definition of traceability along with the components that constitute it for the domain of information risk assessment based on a literature research, and
- A trace model that is built based on a design science approach using examples of different cloud computing architectures to provide support for traceability for TRESPASS.

3

Methodology

This chapter picks up where we left off at Section 1.4 and presents the reader with a detailed look on how we plan to answer our research and sub-research questions using scientific methodologies. A detailed description on how we address each sub-research question to arrive at an answer for our main research question is given systematically.

3.1. Answering SRQ 1

Our first sub-research question demands a definition and specification of the concept of traceability in the domain of cyber risk assessment as an answer. To answer SRQ 1, a literature study was conducted using the methodology presented by Webster and Watson in [218]. We also looked for expert opinions within TRESPASS on the subject. Literature on certain topics was retrieved from reputable search engines (Scopus and Web of Science). Additional articles were searched by both tracing back the cited papers in such articles and tracing forward towards papers that cite the journal papers. Findings are presented in a *concept-centric* manner, meaning all literature on a certain concept is discussed in one section. However, other articles germane to the notion of traceability were also considered (e.g. articles on data provenance in the field of computer science and articles on digital chain of custody in the field of forensics). Documents of standards (e.g. from ISO or IEEE) were perused as well. *Concept matrices* that map important concepts against different articles are constructed for separate disciplines. They conveniently show which articles contribute to which concepts.

Expert opinions were distilled using semi-structured interviews. Semi-structured interview is our data collection method of choice because it fits our purpose of looking for new insights, allowing for openness and flexibility without diverting too far from the original topic. The relevant experts who we interviewed are:

- **Wolter Pieters** (Research supervisor; Delft University of Technology/TUD);
- **Christian Probst** (TRESPASS technical leader; Technical University of Denmark/DTU);
- **Dan Ionita** (TRESPASS researcher; University of Twente/UT);
- **Lorena Montoya** (TRESPASS project manager; UT);
- **Margaret Ford** (TRESPASS researcher; Consult Hyperion/CHYP);
- **Jan Willemson** (WP5 leader; Cybernetica/CYB);
- **Miguel Martins** (WP6 leader; itrust/ITR);
- **Olga Gadyatskaya** (D5.3.3 leader; University of Luxembourg/UL); and
- **Axel Tanner** (Cloud case study researcher; IBM Research Zürich/IBM).

The protocol for the interview is given in Appendix C. Each interview was designed to last for 60 minutes. We analysed the results of these interviews using the methodology proposed by Burnard et al [31], namely *thematic content analysis*. There are essentially three steps in the process, namely:

1. Open coding: Summarising elements that are discussed in the transcript while weeding out deviations ('dross'), by creating coding frameworks. Coding frameworks help with structuring the interview data, by creating categories of ideas to be analysed.
2. Shortlisting all coding frameworks collected in the first step and listing the final coding frameworks. This is done by deleting overlapping frameworks and grouping similar frameworks.
3. Organising the whole dataset by categorising the data according to the final coding frameworks identified in the second step.

To validate and verify the interview results we used an approach called *member checking*, that is returning to the respondents and asking them to validate the analyses. This was done by sending the individual transcripts and analysis results to each respective respondents. Analysis results are presented by means of reporting important findings under separate themes, accompanied by appropriate quotes when necessary. The answer to SRQ1 is a formal definition of traceability as it concerns the activity of cyber risk assessment.

3.2. Answering SRQ 2 and SRQ 3

The trace model for TRESPASS was developed using design science methodology. The phases involved and how we approach them are outlined as follows:

3.2.1. Problem investigation

The goal of this phase is to investigate the problem prior to designing the trace model. To do this, the following activities were done:

Identifying the stakeholders, their expectation of the research and their goals

Possible stakeholders for development of the trace model were identified using the list given by Ian Alexander [7]. These stakeholders either *interact* with the artefact, are within the *immediate environment* of the artefact, are in the *wider environment* of the artefact, or are *involved in the development* of the artefact. Next, we described a small set of goals (G) to be realised with the artefact to be designed [213].

The process to gain insight into the stakeholders and their goals was done through semi-structured interviews. The questions are given in Appendix C, and we analysed its results also using the methodology given in [31]. Goals were formulated based on part of the interview results and subsequently evaluated against criteria proposed by Verschuren & Hartog as follows [213]:

- Clearness;
- Stakeholders' acceptance;
- Feasibility;
- Affordability;
- Opportunity and opportunity cost; and
- Order of priority

Moreover, we referred to documents within the TRESPASS project to gain a complete context. Priority was given to legal documents or work description documents. Deliverables pertinent to *cloud infrastructure*, *dynamic features*, and *extensibility* of the TRESPASS model are not left out.

Identifying the phenomena, their causes and effects and their contributions to the stakeholders' goals

In this activity, we described the phenomena relevant to the problem, recognised their causes and effects and analysed their contributions (regardless of their direction) to the stakeholders' goals. To do this, we prescribed plausible Use Cases in TRESPASS that can be supported via traceability. Likely scenarios were devised, supported with rationales behind them. Further details were discussed with relevant people within TRESPASS.

Finally, we also conducted an ex-post evaluation of the problem investigation stage (after it is carried out) by referring to Verschuren & Hartog [213]:

- "Was the involvement and variety of experts well balanced against the expected impact of the design?"

- "Is *G* sufficiently sharp defined in order to derive the functional requirements (R_f), user requirements (R_u), and contextual requirements (R_c) and to give direction to the next stages in the designing process?"
- "Have standard methodological guidelines for empirical research been followed during the process that led to the formulation of *G*?"

3.2.2. Treatment design

After gaining knowledge of the problem to be treated, we moved on to designing a treatment to cure the problem. This phase comprises the following activities:

Specifying the requirements and context assumptions

A requirement *(R)* is essentially a property that the treatment should possess, or the goal of the treatment. There are three types of requirements [213]:

1. R_f : Functional requirements (functions that the artefact should be able to execute)
2. R_u : User requirements (requirements of future users)
3. R_c : Context requirements (requirements of political, economical, or social nature)

An example of a functional requirement of the TRE$_S$PASS toolchain is that it should be able to generate an attack tree [195]. A user would expect that the TRE$_S$PASS toolchain does not consume too much storage space, which is not necessarily concerned with the functions that the TRE$_S$PASS toolchain can execute and is therefore identified as a user requirement. These requirements were mainly derived from the results of the semi-structured interviews and formulation of possible Use Cases to visualise how users will interact with TRE$_S$PASS. We believe context requirements are irrelevant to the trace model, as the impact of the trace model is very limited to solely TRE$_S$PASS and does not extend beyond that.

Besides requirements, we also formulated assumptions (*A*), which are [213]:

1. A_f : Assumptions about functions to be fulfilled
2. A_u : Assumptions about future users
3. A_c : Assumptions about context

The Use Cases exist to provide assumptions about functions to be fulfilled and assumptions about future users.

These requirements are backed up with *contribution arguments*, or justifications for the choice of requirements. The contribution arguments essentially take the form of:

(Artefact Requirements) x (Context Assumptions) contribute to (Stakeholder Goals)

These requirements were translated into measurable terms into design criteria (*C*). There are also three types of design criteria: C_f, C_u, C_c. The requirements and assumptions were generated by referring to the semi-structured interviews as well as official documents within TRE$_S$PASS, for instance those regarding the extensibility of the TRE$_S$PASS model (D5.3.2), functional requirements of the TRE$_S$PASS model (D6.2.2) and requirements for process integration (D5.1.2). These requirements, assumptions, and design criteria were evaluated with the following litmus tests:

- Empirical validity, reliability, researcher independence and verifiability of the results that lead to the formulation of R_u;
- Whether R_f covers *G* without any errors of omission or errors or commission;
- The logic of the combination of and the relations between R_f *and* R_u;
- The methodological guidelines in deriving R_f, and R_u and translating them to the various design criteria;
- Whether the assumptions are credible enough; and
- Whether the design criteria have been operationally defined, and whether they cover the requirements and match the goals.

Identifying available treatments

We believe there are yet no available treatments to support traceability for the TRE$_S$PASS model. However, we can refer to the trace model defined in the CORAS model [171] as an example. Other notable alternatives are briefly mentioned.

Carrying out the design process

The design process consists of defining structural specifications (S), which are characteristics and aspects that the trace model should have to fulfil both requirements (R) and assumptions (A) [213]. S were defined based on R, A, and C. We defined *processes* that the trace model should be able to execute and sculpted]a *general* design depicting the overall architecture of the trace model. We referred to documents outlining the formal requirements of the TRE$_S$PASS model, for instance data management process (D2.1.2) and policy specification language (D1.2.2). The results of this activity were checked against C_f, C_c, and C_u.

At the end of this phase, we conducted a plan evaluation that seeks to evaluate whether all goals, requirements, assumptions and specifications form a tenacious whole.

3.2.3. Treatment validation

The goal of this phase is to justify that the trace model would satisfy stakeholder goals when implemented in its intended context [221]. Essentially, we are interested in answering two questions:

- *Requirements satisfaction question*: Are all requirements (functional and nonfunctional) satisfied? Why (not)?
- *Sensitivity question*: What will happen in a different context?

The term validation here is used to refer to two distinct activities: verification of requirements satisfaction (first question) and testing the artefact in a different context (second question). Wieringa lists two more knowledge questions in validation research, namely *effect questions* (What effects are being conjured by the interaction between the artefact and the context? Why?) and *trade-off questions* (What will happen with a similar artefact?) [221]. We left out these questions because we do not have a working model to study the effect of the interaction between the artefact and the context, and we did not develop any alternatives of the artefact to be validated. Two approaches were chosen to answer the knowledge questions: Expert opinion and single-case mechanism experiment. Expert opinions were sought to answer the requirement satisfaction question, while single-case mechanism experiment was conducted to answer the sensitivity question.

In general, the verification with experts was aimed towards digging opportunities for implementation of the trace model and collecting opinions on how well the trace model can fulfil the goals and requirements. This was done through means of an interactive validation session. The session was different for each expert and the focus was tailored based on the concerns and suggestions raised by each expert during the semi-structured interview. The experts consulted include those who have participated in the semi-structured interviews during the problem investigation phase, plus another expert who is deeply involved in the development of the user interface and visualisation of TRE$_S$PASS tools. Results of the validation sessions were analysed using content analysis and suggestions for improvements were distinctly highlighted.

Single-case mechanism experiment was selected because we intend to expose the trace model to controlled stimuli in a model of the intended problem context. For this purpose, we selected a case that has a fundamental architectural difference with the case used during the design phase. A detailed protocol on the single-case experiment is given in Appendix D.

4

Defining and specifying traceability in cyber risk assessment

This chapter presents the answer to the first sub-research question, which is a definition and specification of the notion of traceability in the domain of cyber risk assessment. This has been done through a systematic literature study and semi-structured interviews. The semi-structured interviews were used to collect expert opinion about traceability in risk assessment, both in generic and TRESPASS-specific context. This chapter only delineates the opinions on the generic context of risk assessment. The processes of the literature study and semi-structured interviews are outlined in this chapter, followed by the results and the corresponding analyses. We end the chapter with a formal definition and specification of traceability, derived logically from a massive collection of literature and opinions of different experts.

4.1. Procedure

4.1.1. Literature study

The literature study was conducted by employing two main search engines: Web of Science and Scopus. These search engines were chosen for their reputation and for their multi-disciplinary setting. General search terms were used to obtain relevant articles. The initial topic explored was traceability. The search terms used for this topic and the number of results they generated are presented in Table 4.1.

As the initial results from the search term "*traceability*" generated superabundant results, the search was narrowed down by adding an AND operator and a specific domain. The results were then sorted by order of popularity, namely by how many other articles have cited that specific article. This way, less impactful articles could

Table 4.1: Search terms used for *traceability*

Search term	**Web of Science**		**Scopus**	
	Number of results	*Saved results*	*Number of results*	*Saved results*
Traceability	17,525	-	13,634	-
Traceability AND requirements	1,947	50	4,512	50
Traceability AND risk	720	50	1,726	50
Traceability AND review	531	50	3,756	50
Traceability AND food	1,788	50	3,021	50
Traceability AND logistics	212	50	751	50
Traceability AND supply	1,014	50	1,731	50
Traceability AND safety	1,479	50	2,725	50
Traceability AND security	728	50	1,710	50
Traceability AND health	712	50	2,041	50
Traceability AND compliance	660	50	599	50
Traceability AND accountability	61	50	144	50
Traceability AND dependency	120	50	375	50
Traceability AND cloud	83	50	236	50
Traceability AND architecture	510	50	1963	50
Total articles saved		**577**		**509**

be weeded out. However, this approach brings the risk of producing results with little to no relevance and the risk of missing out on recently published but powerful articles. The top 50 results from each search were then saved and compiled into a bigger list. Some overlap occurred: one or more articles appeared multiple times using different search terms. From there, the collective top 100 results were then downloaded for further reading. This was done in both search engines, resulting in a total of 200 articles to be analysed.

The priority of analysis was given in the following order: Articles that both appeared in the top 100 lists from both search engines were given first priority. The following priority was given to the rest of the articles in the list, and subsequently articles cited by publications in the first category. Last priority was given to articles that cite the articles in the first category. All 200 articles were scanned on the basis of abstract and subsequently its content. From a total of 200 articles, 59 were found to be relevant, including the articles that were cited in these works and those who cite them. Relevancy means that these articles contribute formulated definitions or taxonomies that can help in building a framework on traceability. These articles cover a multitude of different domains, from food sector to metrology. Articles on practical applications rarely make the list. Besides articles that pertain to traceability, we also conducted systematic searches on other topics that might potentially contribute important aspects to our definition. These topics are *data provenance*, *digital chain of custody*, and *audit log*.

The concept matrices are shown in Appendix E. In this chapter, two sections are exclusively devoted to discuss works in the domains of *food sector* and *requirements engineering* respectively, as prominent works on traceability are conceived from researches in these two areas. Results from other domains such as logistics, clinical chemistry and cryptography are discussed collectively in one section. Results on other search terms, such as data provenance and digital chain of custody, are shown separately.

4.1.2. Semi-structured interviews

Expert opinions were distilled using semi-structured interviews. We did the interviews with all nine experts mentioned in Chapter 3. However, not all parts of the interviews are relevant for answering SRQ1. We only consider the responses to the first and the third topic, namely *traceability* and *traceability and dynamic risk assessment*. The interviews were conducted within the time range of February 29, 2016 until March 18, 2016. For confidentiality reasons, the transcripts of the interviews will not be made public, the respondents' names are anonymised, and answers are not directly attached to specific respondents. In this chapter, we only go to the length of discussing the main ideas given by the experts. Snippets of quoted sentences will be presented when necessary. The findings shall be supported with a discussion section and closed with a concluding section.

4.2. Results from literature study

In this section we discuss our findings on state-of-the-art literature across multiple domains. The concept matrices are presented in Appendix E and are categorised per domain.

4.2.1. Traceability in food sector

The notion of traceability in food sector spans across multiple industries, such as meat industry, coffee industry, and genetically modified organisms (GMOs). Several concepts hold a universal definition among these industries, but others are interpreted differently. The identified articles have appeared in leading journals such as *Journal of Agricultural and Food Chemistry*, *Meat Science*, and *Trends in Food Science & Technology*. The literature study resulted in more than 60 important concepts with regards to traceability in food sector. Several of these important concepts will be addressed in this section. We begin with *tracing* and *tracking* as a few of the most commonly found concepts. The basic idea of tracing in food sector is being able to follow a *product* (food/feed/substance) through the *history* of different *stages* in production and distribution processes (e.g. [58]), while tracking amounts to continuous *monitoring* from upstream (harvesting from *source*) towards downstream (sales to customers) (e.g. [127]). Traceability in food industry is most commonly found in the form of *product traceability*, be it in the form of feed, substance, or even the animal itself (e.g. [126] [127]). Traceability in food sector is expected to be present at all stages of the food supply chain (production, processing, and distribution). Another recurring concept is *safety*, which is one of the goals of traceability in different subdomains in food sector (e.g. GMOs [126] and meat industry [78]). Different forms of *identification* are present to establish traceability, one of which is labeling [170]. An example of such labels is a country of origin label, which shows where a certain food product was grown and harvested.

An important distinction is made between *voluntary traceability* and *mandatory traceability* [210]. Both forms of traceability are supported by their own respective regulations, and are usually industry-specific [67]. As the name implies, voluntary traceability is optional, while mandatory traceability is obligatory. The former seeks to correct market failures due to asymmetric or imperfect information (from the customer side) regarding negative attributes

of a certain product [77] and the latter aims mainly at product differentiation and is more directed towards attaining desirable product attributes. We also observe several important classifications of data. These include *static data* versus *dynamic data*, as well as *mandatory data* versus *optional data* [58]. Static data record product features that cannot change, such as country of origin (COO) and expiry date. Dynamic features that might change are captured by dynamic data, for example batch number and order ID. Mandatory data have to be collected from all parties involved in the supply chain and needs to be communicated, while optional data do not play an equally important role in the supply chain.

Finally, Hobbs et al [77] mention that traceability systems can differ in *breadth*, *depth*, and *precision*. These terms respectively refer to the amount of recorded information, the extent of tracking and tracing activities, and the degree to which a product's movement can be accurately pinpointed. [58] divides traceability information flow in two types: *one step up-one step down flow* and *aggregated information flow*. While the former allows consumers to receive only necessary information regarding basic features of products (e.g. origin and quality), the latter provides consumers immediate access to information during *all* stages of production. This is also known as *from farm to fork* flow. However, the EC Regulation 178/2002 on food sector traceability recommends the one step up-one step down model [58].

4.2.2. Traceability in requirements engineering

Requirements engineering is a sub-discipline within the larger domain of software engineering, which studies the relationship between software artefacts and their relationship to stakeholder requirements [41] [145]. This concept has been explored since the 1970s and has undergone major reinterpretation [99]. As systems grow to become more complex and interdependent, requirements engineering has widened its scope from individual systems towards software-intensive ecosystems. The identified articles on this domain have appeared in leading journals such as *IEEE Transactions on Software Engineering*. The literatures in requirements engineering reveal that traceability has been explored for more than two decades [70], showing that this concept has received much attention and has been recognised as an integral part of system development. Concepts in food sector such as *tracing* and *origin* recur in requirements engineering, as well as *backward traceability* and *forward traceability*. Backward traceability traces back a requirement from deployment back to its origins [70] and seeks to support root-cause analysis of the design of an architecture. Forward traceability seeks to facilitate future referencing of a specific requirement [70] and conduct an impact analysis of the architecture design [178]. The importance of certain concepts such as *trace links* and *stakeholder requirements* are raised, which lie at the very heart of requirements engineering [145]. Trace links enable both forward and backward traceability in a software design. We also came across attempts to build devoted methods of traceability (e.g. *event-based traceability* [41] and *model-driven traceability* [5]). Event-based traceability allows notifications to be sent to dependent artefacts whenever requirements are altered, while model-driven traceability allows for propagation of small changes towards relevant parts of a target model by re-executing transformations. Other works present effort to dissect traceability into meaningful taxonomies, such as *pre-requirements traceability* (before requirements specifications are included) and *post-requirements traceability* (after requirements specifications are included) [70], or *requirements-to-object model* (RTOM) and *inter-requirement* traceability rule (IREQ) [173]. RTOM rules create traceability relations between requirements and elements of the object model, and IREQ rules create traceability relations between different parts of a requirement.

Some intriguing concepts that may bear similarities with cyber risk assessment are for example *impact analysis*, *dependencies*, and *evolutionary change*. In this context, impact analysis assesses the effects of changes (mostly in documentations) to the source code [12]. For cyber risk assessment, this is comparable to how changes in the system would affect the model of reality created for the risk assessment process. Dependencies characterise interrelationships between artefacts during development process [54]. In a complex system, tight dependencies arise between the system components. These dependencies bring knock-on effects in a cyber risk assessment activity, in the sense that these system components should not be seen in isolation, but rather holistically. Couplings between components propagate even the smallest changes to the final risk picture being produced. Finally, evolutionary change amounts to the constantly evolving requirements in a complex system [145]. This is analogous to the concept of *continuous evolution perspective* of change in CORAS [117] except for the fact that the latter type is deemed as foreseeable, while requirements might change radically.

4.2.3. Traceability in other domains

Other than in food sector or requirements engineering, traceability as a concept is also utilised in different domains. The literature study revealed that domains such as cryptography and metrology are also familiar with traceability, albeit the difference in interpretation (for instance [18] and [159]). The articles that were identified in these domains were also published in high-impact journals, such as *Journal of Supply Chain Management* and *IEEE Transactions*

on Vehicular Technology. Within the various domains that fall beyond food sector and requirements engineering, we re-encounter concepts that have been prominently used in the two aforementioned domain such as *tracing* and *tracking*, *chain traceability*, and *breadth*, *depth*, and *precision*. However, we did not identify any prominently featured concepts that might apply to risk assessment other than the well-known concept of *tracing*. The pinpointed concepts are mostly context-specific and rather non-transferable to the domain of cyber risk assessment.

4.2.4. Standards on traceability

As traceability is widely recognised in different domains, there exists a variety of standards and regulations for traceability. Various standards' guidelines and official documents for regulations were also consulted to enrich the results gained from literature study, which mostly produced scientific publications. The various definitions encountered in different standards documents mostly revolve around the idea of tracing and tracking, either separately or collectively. Both tracing and tracking appear in for instance, the IEEE definition for traceability in requirements engineering, or in GSI's definition for traceability in supply chain. However, traceability in metrology as defined by JCGM consists of only the tracing dimension, and Codex Alimentarius posits traceability as only having a tracking function in food sector.

4.2.5. Other search terms: Data provenance, digital chain of custody, and audit log

Data provenance

To seek impactful articles on data provenance, we once more employed Web of Science and Scopus. The initial search returned ample results, hence we reduced the scope to only include articles on data provenance in computer science. A more reasonable number of results turned up, and we made a short list of these articles by limiting the analysis to only the top 50 articles from both search engines. Further analysis brought the final list down to 28 meaningful articles. The notion of provenance is prevalently known in the domain of art. The Merriam-Webster dictionary defines provenance as "the history of ownership of a valued object or work of art or literature". However, the concept of provenance has been adopted for use in the domain of computer science, particularly in the context of the Semantic Web. The W3C Provenance Incubator Group proposes the following working definition for provenance for the web:

> *"Provenance of a resource is a record that describes entities and processes involved in producing and delivering or otherwise influencing that resource. Provenance provides a critical foundation for assessing authenticity, enabling trust, and allowing reproducibility. Provenance assertions are a form of contextual metadata and can themselves become important records with their own provenance."* [216]

Provenance has other aliases, including *lineage*, *pedigree*, or *audit trail* [49], *derivation history*, *data set dependence*, *filiation*, or data genealogy [25]. It has to do with the *derivation* of an item or a *data product* from its source [25] [168]. According to Moreau [128], provenance can be equated with a *logbook*, which contains each and every step leading to the derivation of a certain result.

Closely related to the concept of *provenance*, there is the concept of *workflow*. There is a subtle difference between the two concepts. [25] Bose and Frew argue that workflow is *prospective* and prescribes plans for processing, while provenance is of *retrospective* nature, expounding how the data has been derived after the actual processing has taken place.

There exist multiple types of classification for provenance. For example, Simmhan et al [168] mention the difference between *data-oriented* provenance, where information is collected for the data product. The other side of the story is *process-oriented* provenance, where provenance information is amassed specifically for the derivation processes. A more popular classification is based on the *granularity* of the provenance data. Two categories of provenance exist: *coarse-grain* provenance and *fine-grain* provenance. In the former, the complete derivation history is kept, while the latter regards only part of the resulting data product [28]. Coarse-grain provenance is also known as workflow.

Finally, another essential distinction is made within fine-grain provenance, namely *where-provenance* and *why-provenance* [128]. Where-provenance pinpoints the location where a data product was copied from in the source, while why-provenance seeks to justify the rationale of why a certain query result is present in the output [28] [128]. A further distinction is made by including *how-provenance*, which explains how an output tuple is derived [36] [128].

Digital chain of custody

Another concept that we deem related to traceability is chain of custody, particularly digital chain of custody that commonly occurs during digital forensics process. However, we only came across limited number of articles that

contain appreciable concepts. A commonly cited definition of chain of custody is the one defined by The U.S. National Institute of Justice, which is "a process used to maintain and document the chronological history of the evidence." [27] [66]. Chain of custody is deemed important to preserve the tactfulness of evidence (*integrity*) in a digital forensics process. A customary way to establish integrity is by applying *digital signature* on the evidence [46]. Simply put, chain of custody has a lot to do with evidence handling during an investigation process.

Several authors have reiterated the importance of applying the classic 5WlH formula (What, where, who, when, why, and how) during digital forensic investigation process. A rigorous chain of custody must be able to answer all these questions (e.g. [16] [27] [45]). Schäler et al [160] propose that security requirements such as *confidentiality*, *authenticity*, and *reliability* must also apply to chain to custody. A link between chain of custody in forensic and data provenance in databases was drawn in the same literature.

Audit log

Ultimately, we also looked for articles that discuss the concept of audit log. Audit log involves monitoring computer-generated information from the so-called *event log* regarding a *process instance* or a case that has multiple descriptive attributes, such as *activity* (what), a *performer* or an *originator* (who), a *timestamp* (when) and other attributes [156] [180] [204]. Audit logs are performed for different purposes, such as *conformance checking* (finding a match between recorded events that represent the real process and the process model in an organisation [154]), or *process discovery* (automatic construction of a process model based on event logs [155]). Another rationale behind audit logging is to comply with standards, such as HIPAA (Health Information Portability and Accountability Act) in the healthcare sector [113].

We identified several techniques to perform audit logging, such as *process mining* which enables knowledge extraction from event logs. An instance of this is the ProM framework [203]. Another possible technique is the *Blind-Aggregate-Forward logging scheme*, which places extra emphasis on *forward security* (ensuring past information is not compromised based on present information) [18].

Similar to chain of custody, we identify that to be useful, audit logs need to contain information such as:

- What is being done, or a process step. This can fall under different terms, such as *activity* [156], *task* [206], or *event* [180]. The aggregates of these actions also come in different parlances, such as *traces* [156] or *session* [180];
- Who performed the action, or the *originator* [204]/ *agent* [206]/ *performer* [156]
- When the action was performed, which is usually supported by *timestamps* [156]; and
- An identifier (ID) that uniquely identifies each process step [207].

4.2.6. Conclusions

The rigorous literature study helped us in identifying important themes on traceability from different disciplines. The concepts of tracing and tracking can be found in various domains. In food sector, traceability concerns mainly food product and covers different stages starting from the harvest phase up to the point where the food product reaches end consumer. Requirements in a software development project can either be traced backward or tracked forwards. Traceability in this domain can serve multiple purposes, such as impact analysis, showing dependencies, or representing evolutionary change. Different traceability standards posit different definitions of traceability, although to a certain extent they essentially deal with tracing and tracking. Data provenance can come in different levels of granularity (coarse-grain or fine-grain) and can take up multiple forms, such as why-provenance and where-provenance. Finally, articles on digital chain of custody and audit log stress the importance of 5WlH information (in a forensic investigation process or in an audit log) to facilitate meaningful analysis.

4.3. Results from semi-structured interviews

In the semi-structured interview, there are two question items that were formulated specifically to probe the experts' thoughts on traceability and risk assessment. The first question, which is more general in nature, is posed at the very beginning of the interview. The second question combines the notions of traceability and *dynamic* risk assessment, and attempts to see how experts think these two notions are interlinked. We heard a lot of varying responses, which can be attributed to several factors, such as the background of the expertise of each expert, past projects they have worked on, and their current role in TRESPASS. Some opinions can be directly translated into requirements for traceability, which will be demystified more in Chapter 5.

Table 4.2 presents the initial coding frameworks for these two topics. The first digit of the code indicates the associated topic in the interview (e.g: Item 1.3 is associated with the first interview topic). The sequence indicator (second digit) does not necessarily show the order of the respondents.

Table 4.2: Initial coding frameworks for traceability in risk assessment

Code	Interview transcript	Initial coding framework
	Interviewer: "How would you define traceability in a risk assessment process? What should it consist of?"	
1.1	"Being able to trace back the individual inputs that led to that risk assessment."	Ability to trace back to inputs
1.2	"Traceability is very important because there's a lot of errors that can be introduced in the model... and you should be able to account for that and be able to identify where the source of that error might be..."	Identifying the source of errors in analysis
1.3	"... but not just errors but also the sensitivity of the analysis results."	Sensitivity analysis
1.4	"I think that the manual mode of traceability is linking components to risk, especially if you're doing a model-based risk assessment. The higher level of traceability would be storing the arguments by which each risk applies to a certain component, so the rationale behind that risk... an even higher level of traceability would even have to involve the security decisions themselves, the mitigation decisions. So first level is relating assets to risk, second level would be explaining why those risks are there and why they affect those particular assets, and then a third level would be explaining which mitigations relate to those particular risks and why they reduce them."	Multiple levels and purposes: (1) Linking components to risk; (2) Rationale behind the risk; and (3) Explanation behind risk mitigation strategies
1.5	"The ability to generate results and then to return to those results to verify that the inputs are consistent with the outputs... to be able to return to the inputs from the outputs, to be able to understand where the outputs come from."	Verification of outputs
1.6	"Throughout the project we've been discussing the value of expert opinion and how very variable it can be and how to quantify that, and in that sense traceability is very important in terms of the source of your data, and at the same time handling that in an appropriately sensitive way if you're working particularly with industrial partner is extremely difficult." "You've got the question of calibration, so you'll get maybe four experts, they all have very valid opinions, very different kinds of experience will give you very different answers. And you then have the challenge of turning that into something reasonably meaningful in terms of measurement." "There are different ways to input on the different opinions by different experts but also the extent to which those experts will make commitment to their particular opinion in different circumstances."	Handling expert opinion: calibration, reliability
1.7	"The implicit trace/tracing/traceability is always there as soon as you have some analysis based on a system model, then you can immediately call any changes, traces if you want." "I'm sure implicitly it's always there, well because implicit meaning the human has to use his own intelligence to see the trace, I mean the human can put side by side two different models and put side by side two analyses and see what's different."	Implicitly present in risk analysis
1.8	"I think it should consist of linking between input and outputs most generally so that when the input is changed you might be able to identify in the output results where it relied, where this came from, how it influences in my change, the new output."	Link between input and output
1.9	"In risk analysis, it's more about tracing risk assessment, risk values, risk calculations to initial data that these calculations are based on. So if you have certain quantitative information that is used in your risk analysis, then ideally you'd like to know, where in risk assessment this information is used, and therefore also when something changes in this initial information, for example if a new vulnerability is discovered or something, then you can figure out which part of your risk assessment would therefore have to be updated. It's much more about data traceability, if you understand, in the ground of requirements engineering."	Data traceability: pinpointing data in risk assessment
1.10	"This is sort of an approach that will help us to manage links between various models and different parts of the process. ...merge different aspects, then establishing traceability links between all these models to something explicit."	Managing links and models in a process
1.11	"...a trace from parts of the model to the answer or to the result of the risk assessment and also the other way around. Once you have the risk assessment that you are able to go back to see which parts of the model actually have influence to the result of your risk assessment."	Linking model components to risks
1.12	"I think it's also interesting to trace how changes of the model influence changes in the risk assessment."	Propagating changes

Table 4.2 Initial coding frameworks for traceability in risk assessment (continued)

Code	Interview transcript	Initial coding framework
	Interviewer: "How do you see the role of traceability in facilitating a dynamic risk assessment? As in, how do you see traceability could help with keeping the validity of a risk picture after changes happened?"	
3.1	"Well it would depend on your... threat spectrum I think. Only by looking at emerging threats you could... you could decide for example on how often to update the risk assessment. It's always a tricky issue because it's all... I mean you can say 'Oh ideally I want it every week' but who's going to give you the updated data?"	Dependant on threat spectrum
3.2	"When you analyse impact then you have the concept of acceptable risk comes up. And there you see it with insurance companies. Insurance companies will say, for that amount of impact we don't care, we accept it. We can pay it. We can pay out. So that threshold is very very important and I think that threshold you need to know in order to decide, make a decision on a dynamic risk assessment. How often do you need it? So that's a big question mark."	Connected to the concept of acceptable risk
3.3	"For me the traceability will be most important to do a sensitivity analysis to see what happens when you do slight changes in the values and of course you're gonna tell me that's a dynamic risk assessment... but I think for me that is the most important part. How stable are those twenty threats you have. Is this one that you identified as being the most important one, given an input change of something very very small in the input value over a year going to completely change the ranking and send you the first threat to number fifteen."	Sensitivity analysis
3.4	"The robustness of your final ranking of threats. So if you give me a whole spectrum of applications of traceability then I would tell you that's the one I'm going to put my money on."	Robustness of analysis results
3.5	"I'm in the social data side, and I know those things can change... the picture can change dramatically and it's also well... even if you... where are the attacker profiles? That can change in a nutshell. That's so highly volatile. So that can also be very dynamic."	Data volatility
3.6	"You want to able to store the changes that you do to the model. So exactly, I think it's important in order to maintain... in order to be able to have a dynamic model, to have functionality that allows you to somehow keep a log of the changes you do. So whether these changes were done with the purpose of just sensitivity analysis which is seeing what a change does, or whether they were done as an update. Let's say another year has passed and you want to update the model with new improvements, still you should be able store the old model just to see, so you can quickly see what changed."	Model storage and changelogging
3.7	"I would see you've got the short term things and you've got the long term things or probably quite a lot in between. But if you look at the short term things, it's everything I've been talking about already. So making sure when you get your results, you know where they come from. You know which parts you can adjust, and you can meaningfully compare the results as you adjust things. So in that respect, that's how I would read dynamic in that environment. Then you've got the long term things which are much harder to deal with... emerging threats, organisational changes, and traceability is more crucial then. You've got at least two different influences there; you've got the elements that will change and you know for sure they will change over time, and you've got the question of how frequently you should revisit them, and so having some kind of... just something basic that's not far off a reminder system wouldn't be a bad thing. Something that incorporates that element of time and possibly according to particular triggers."	Customised, periodic trigger mechanism
3.8	"Something that comes in here, which is very visible in the industry at the moment is the question of... sharing data, sharing information on attacks, and there are different schemes aimed at encouraging people within similar industry to share relatively sensitive data but in a reasonably anonymised way, and I see it as very important in terms of the dynamic approach, so where you have an industry body, they can perhaps keep a watch on growing areas of vulnerability across the industry and encourage particular attention in that area."	Enabling industry-specific data sharing
3.9	"... if we receive some kind of notification from the manufacturer, then it would pinpoint you to revisit this part of the model, and that way you could kind of push in the new information relating to that particular part without having to mark about too much and you'll have to rerun the calculations but hopefully you won't have to mess with the rest of the model too much." "It's making everything a little bit more modular, rather than a sort of a big big picture."	Modularising risk analysis process
3.10	"In this sense, and the reason why we do this is exactly the thing that you call traceability, but you want to understand what will be changed in the analysis, or the backwards if I have the analysis what should I change in the model in order to improve my analysis results to be more favorable to me, right? So I mean these are two sides of one coin, I would say."	Backward and forward analysis

Table 4.2 Initial coding frameworks for traceability in risk assessment (continued)

Code	Interview transcript	Initial coding framework
3.11	"...from the technical side, you would expect you can quickly, immediately do an analysis on an exact level basically. All the social data enters of course a much lower time space...it wouldn't make sense to have a real-time analysis if the outcomes are something which influence things on the scale of months." "Simply doesn't really make sense to aim for real-time things." "So input data, changing analysis results, you look at your analysis results, you run like a what-if scenario by changing the model by hand, and reanalyse. That's obviously much slower cycle."	What-if scenario instead of real-time analysis
3.12	""It's a... as a bookkeeper or something like this. So you can keep the record of what changed to arrive at such a risk scenario or risk situation."	Bookkeeper role
3.13	"The idea with traceability would then be that if some of these data changes, so there's a new vulnerability, or whatever, then you would also know which parts of the risk assessment to update. So that means rather than waiting for the next regular risk assessment exercise, you could update the risk assessment on the fly, assuming that it would not affect all of the risk assessment, because then you probably have to do it all over again anyway." "Ideally you wouldn't need to redo the whole assessment, that's the idea because you keep track of these links, and it should be ideally possible to quickly update relevant parts of the risk assessment without having to redo the whole exercise..."	Facilitating on-the-fly risk assessment
3.14	"Well if it's implemented in the right way, then actually this is what can enable this validity, so because if you can seamlessly propagate the changes, then up to the point where the stakeholder needs to make a decision based on the analysis result, you can maintain everything kind of in the current state."	Propagating changes
3.15	"What I think would make sense is to have a distinction between kind of internal changes, so something that you decide to modify and external changes or something that happens without you agreeing to it or not. So like a new vulnerability is discovered or a new type of attacker emerges. So I think these two types of changes need to be distinguished."	Distinguishing changes
3.16	"...a way to perform the risk assessment when something in the model has been changing. So basically, traceability would allow you to see that, okay this part of the model has a certain influence to our current risk assessment result" "And then you hopefully would be able to select those that depend on the changed parts of the model to recompute something." "So incremental analysis is quite difficult, I don't think... you'll not be able to add process in TRESPASS but on the other hand these are standard techniques, so once you have an incremental analysis you could just apply it to the model."	Incremental analysis of changes

Table 4.3 assembles the different coding frameworks identified in the previous step and shows the final coding frameworks. One obvious limitation of this research is that it is only carried out by one researcher, which consequently brings up the issue of lone researcher bias in analysing the content. We are inhibited from establishing intercoder reliability, which in qualitative researches can show the objectivity of a content analysis. To bring down subjectivity to a minimum level, the coding frameworks are put under scrutiny by the thesis supervisor.

Table 4.3: Final coding frameworks for traceability in risk assessment

Final coding framework	Initial coding framework
Input-output relationship	Ability to trace back to inputs
	Sensitivity analysis
	Verification of outputs
	Link between input and output
	Explanation behind risk mitigation strategies
Data management	Identifying the source of errors in analysis
	Handling expert opinion: calibration, reliability
	Data volatility
	Enabling industry-specific data sharing
	Data traceability: pinpointing data in risk assessment
	Connected to the concept of acceptable risk
Change management	Propagating changes
	Distinguishing changes
	Dependent on threat spectrum
Risk assessment process management	Managing links and models in a process
	Model storage and changelogging
	Customised, periodic trigger mechanism
	Modularising risk analysis process
	Backward and forward analysis
	What-if scenario instead of real-time analysis
	Bookkeeper role
	Facilitating on-the-fly risk assessment
	Implicitly present in risk analysis
	Robustness of analysis results
	Linking model components to risks
	Rationale behind the risks
	Incremental analysis of changes

We identified four prominent ideas on traceability and risk assessment based on the various opinions from different experts. These ideas are given in separate sections as follows.

4.3.1. Input-output relationship

Within this umbrella term, experts construe that traceability should be able to help establish a link between input and output. The purposes of these links, however, can differ. An example of a possible goal is tracing (items 1.1 and 1.5). Another possible goal is to perform a sensitivity analysis, that is to check how much change will be incurred in the output given a certain measure of change in the input (items 1.8 and 3.3). This implies that according to the experts, building a link between input and output is a necessary requirement in devising traceability. Using general risk assessment terms, input is It is also believed that the output should encompass risk mitigation (generally, risk treatment) decisions produced (item 1.4).

4.3.2. Data management

While the previous topic concerns the need for a connectivity between input and output, a bigger overarching topic regarding managing the *input* needed for the assessment is also identified. We name this *data management*, which

comes with a multifaceted interpretation, including how to identify errors in the analysis and how to pinpoint the behaviour of certain data during risk assessment process (items 1.2 and 1.9). Regarding data sources, traceability is viewed to have a potential role in managing the volatility of certain data types, such as expert opinions (items 1.6 and 3.5). Finally, traceability is thought as an enabler of industry-specific data sharing (item 3.8). We conclude that experts believe that traceability may have a wide range of uses which stretch beyond the risk assessment process.

4.3.3. Change management

According to some of the experts, the relation of traceability and dynamic risk management has a lot to do with managing changes. Traceability depends a lot on the velocity of changes happening within the environment of interest (item 3.1). Within this context, traceability is expected to be able to show how changes propagate into the whole pipeline of risk assessment process, such that the results accurately reflect the reality. Not only that, but traceability should be able to explicitly show whether the changes are induced internally (from the organisation) or externally (from the environment) (items 3.14 and 3.15).

4.3.4. Risk assessment process management

The experts posit that traceability in general is beneficial to the process of risk assessment. Some experts break down the function into more specific characterisation, such as storing different versions of model, linking model components to their respective risks (in a model-based risk assessment) and generating logs of changes happening to the model (items 1.4 and 3.6). Some experts also wish to have traceability portrayed as an automated support within the risk assessment process, supporting an on-the-fly risk assessment process (items 3.7 and 3.13). This also has to do with modularising the process (items 3.9 and 3.13). Other experts envisage even more sophisticated functionalities, such as explicit link of specific rationales to certain risks and explanation behind risk mitigation strategies as an output of the risk assessment process (item 1.4).

4.4. Specifying traceability in cyber risk assessment

Based on the literature study in various domains and the extracted opinions, we highlight elements that need to be captured by traceability when it comes to cyber risk assessment. They are selected on the basis of frequency (the concepts commonly recur in assorted domains in the literature and were also mentioned frequently during the interview). These elements or requirements include:

- Traceability needs to support a connection between the input and output of a risk assessment process. Moreover, this connection needs to be present in two directions: backward and forward. Traceability then needs to be able to support both tracking or a prospective analysis from the input towards the output and tracing or a retrospective analysis from the output back to the input. Trace links that show these connections will have to be present.
- With regards to input, traceability must be able to support and handle different data types that are fed in the risk assessment process. More 'static' data that do not frequently change should be treated differently from 'dynamic' data that might be highly volatile. Quantitative data that deal with risk calculations should be set apart from qualitative data such as expert opinions. Traceability, however, should be able to help risk assessors trace back these different types of data back to their respective sources and show how these different types of data will affect the output of the risk assessment process.
- With regards to output, traceability should be able to point out and trace back errors in the risk analysis results all the way to the data source(s) that contribute(s) to the errors.
- Traceability needs to be able to help risk assessors in performing an impact analysis or sensitivity analysis by showing how much the risk analysis result will change given a certain measure of change in the input.
- Traceability should be able to help risk assessors distinguish different changes that might happen to the model (e.g. internal changes versus external change, or radical change versus continuous change) by showing how different types of change will affect the risk analysis result differently.
- Traceability should be able to help risk assessors explicitly see dependencies in the system model and dependencies between different risk assessment process through the presence of trace links.
- Traceability should be able to generate a detailed log of change process in a recurring risk assessment process for an identical target. A coarse-grain log might be kept at the back end of the risk assessment process, while a fine-grain log can be brought to the front end to inform the users.

- At the bare minimum, traceability should be able to link components of the system model (e.g. assets) with risks that are relevant to them.
- Traceability should come with an automated support to flag changes (trigger mechanism) or a reminder system for relevant issues such as emerging threats or new vulnerabilities that require attention over time. This trigger mechanism should be able to highlight elements of the system model that will potentially be impacted by the changes. Traceability should be able to support a rapid risk assessment as opposed to a complete reassessment.
- Traceability log data should contain essential information such as a unique identifier of the change instance, *who* performed the changes, *what* entities were changed, *when* the changes were made, *why* changes were made (arguments behind the changes), and the description of the changes. These data are necessary to perform meaningful checks or audits in the future.

4.5. Defining traceability in cyber risk assessment

Based on the requirements identified, we propose the following definition for traceability in cyber risk assessment:

> *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.*

5

Problem investigation

This chapter marks the start of TRESPASS-specific discussion in this thesis. It discusses the first step of the design science methodology which is used to specify the trace model. In this chapter, we explain the stakeholders that are involved with the trace model (from conception to use) in TRESPASS, along with their expectations and goals. Part of this discussion will be derived from the semi-structured interviews mentioned previously. In the latter half of this chapter, we formulate two different scenarios to explain the phenomena that we will explore further. The outcome of this chapter serves as building blocks for Chapter 6, in which we formulate different requirements and assumptions necessary for building the trace model.

5.1. Identifying the stakeholders, their expectation of the research and their goals

To identify the stakeholders, we refer to the onion model of stakeholder taxonomy by Alexander [7]. This taxonomy was chosen because it was proposed in a similar context to our research (i.e. system development/ requirements engineering). The onion model is given in Figure 5.1.

The onion model includes a wealth of potential stakeholders that could be involved in a system development project, according to different subsets of entities ('Circles'). Each Circle contains 'Slots' which are classes of 'Roles'. A Role is a class of stakeholder with a unique relationship to the product being developed. We give a short explanation on different Roles and Slots on the four circles of the onion model as follows [7]:

1. **The Product** or **the Kit**: the artefact being developed.
2. **The System**: The Product, its human operators, and its operating procedures. There are three types of operator Slots, including:
    - *Normal Operators*: Those who give commands and monitor Product outputs. They interact directly with other Operators and Functional Beneficiaries.
    - *Maintenance Operators*: Those whose roles are to maintain the Product. They interact with the Product and with Normal Operators. In an outsourcing context, maintenance operators may include contractors or service providers.
    - *Operational Support*: Those who provide support to Normal Operators of the Product on how to operate it. Simply put, they are 'maintenance' for involved humans.
3. **The Containing System**: Our System and its human beneficiaries. Beneficiaries can be further differentiated into:
    - *Functional Beneficiary*: Those who benefit from the results provided by the Product. They interact with Operators by giving them instructions and receiving information from Our System.
    - *Interfacing System*: Roles which are responsible for adjacent systems that have interfaces with the Product. They interact with Operators and the Product.
    - *Purchaser*: Those who are responsible for having the Product developed, such as a Product Manager for a mass-market product.

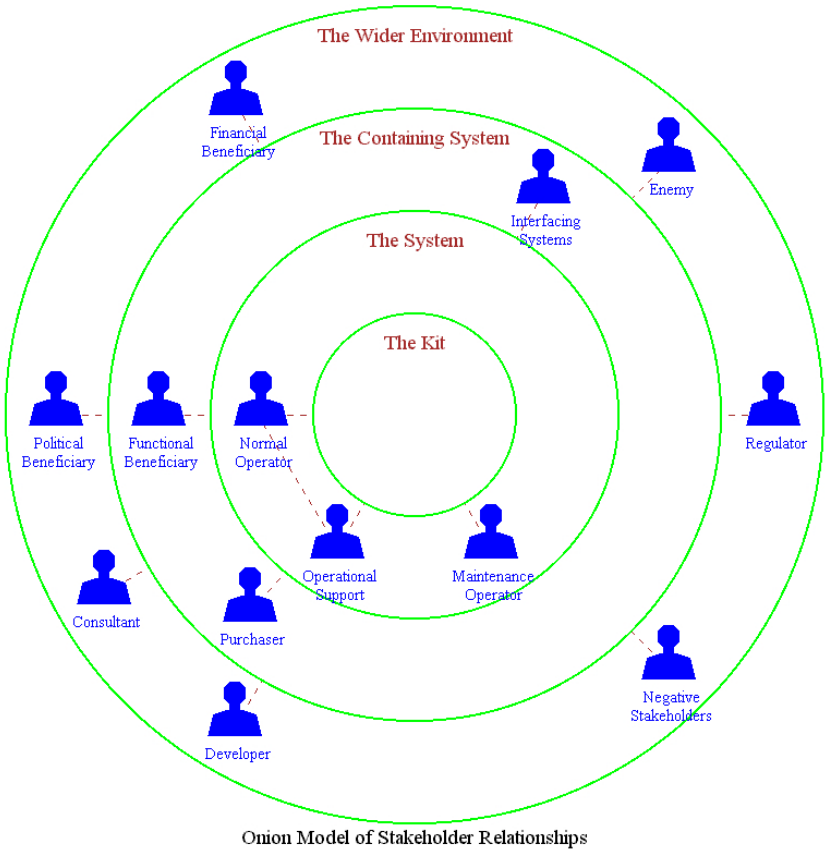


Figure 5.1: Onion model of stakeholder relationships

Source: [7]

- *Product Champion (Sponsor)*: Those who pioneer the development of the Product.

4. **The Wider Environment**: The Containing System and other relevant stakeholders, including:
   - *Negative Stakeholder*: Any roles that could be tarnished by the Product, or conversely try to harm the Product.
   - *Political Beneficiary*: Any roles that can benefit from the Product's success (e.g. in terms of power or influence).
   - *Financial Beneficiary*: Any roles that can gain financial benefits from the Product's success.
   - *Regulator*: Any roles that are responsible for regulating the different aspects of the Product (e.g. legal, quality, safety etc.)
   - *Developer*: Any roles involved directly in the development of the Product.
   - *Consultant*: Any roles from outside the organisation that support different aspects of the Product development.
   - *Supplier*: Roles that are involved in the provisioning and procurement of the Product components.

Our preliminary indication of the stakeholders includes the European Commission as the principal source of financing of the TRESPASS project, members of the TRESPASS consortium directly involved in WP5, as well as intended end-users. The Product in our case is the trace model. To consider expectations of the different stakeholders of the trace model, we again resort to the responses of semi-structured interviews, in particular the TRESPASS-specific question: *What do you think are the opportunities for traceability in TRESPASS? How do you think the traceability feature can bring added value to TRESPASS?*. Inputs from both researchers and industry practitioners were treated as empirical data to paint a picture of stakeholders' expectation of the trace model. The initial coding frameworks and final coding frameworks are presented respectively in Table 5.1 and Table 5.2.

Table 5.1: Initial coding frameworks for opportunities of traceability in TRE$_S$PASS

Code	Interview transcript	Initial coding framework
	Interviewer: "What do you think are the opportunities for traceability in TRE$_S$PASS? How do you think traceability feature can bring added value to TRE$_S$PASS?"	
2a.1	"I suppose even on the attack tree... you can use traceability to find whether trace back the formulas that were used in the nodes of the attack tree"	Trace back risk calculation
2a.3	"With any decision support system, traceability is important because you want to be able to defend your decisions...especially in security where if things go right then nothing happens but if things go wrong, that's when you have to be able to defend your assessment and your decisions." "It should give the decision makers support for deciding to mitigate one risk and not the other, and traceability is part of this support...The basic level of traceability, you need to be able to show in case the risks you decided not to mitigate do happen...It should provide data or... well data to support comparing one risk to the other, okay this is more serious, this is less serious, and this should be mitigated, and this should not be mitigated"	Provide support for security decisions
2a.4	"It's the links between each of those... the means to be able to get back to the previous stage and to understand where the results come from and through that to be able to adjust the previous stage so that you get new results."	Understanding the connection between different TRE$_S$PASS processes
2a.5	"You probably need some kind of process whereby the modelling decisions are recorded." "It's between data collection and model creation...My feeling is quite a lot of the model creation is reasonably manual, it's supported by tools but it requires... it depends a lot on professional expertise and experience. That's necessary, I don't think there's any other way, but if there were a means of recording the underlying decision making process behind that, that could add a huge amount of value."	Understanding the modelling decisions in the model creation stage
2a.6	"Equally the transition from the model creation stage to the analysis stage. Model creation is quite clean, you have the TRE$_S$PASS model come out there, and it's very well-defined. The analysis stage... potentially you've got lots and lots of different analysis tools there. And that is mostly I'm guessing where the iterative process will come in, because you will feed in the model into associated values to the analysis to perhaps just one analysis tool initially. But you'll get some kind of result back and chances are you'll look at it and think, 'Hmm okay, what does that actually mean?' And you may well want to feed in either new values, so that's effectively at the data collection stage, or you may think that a different tool will actually give you a more meaningful result, or you may look at the analysis result and think, 'Oh God the model's not looking quite right' and actually have to readjust the model. So in that sense at the analysis stage, traceability is crucial..."	Supporting iterative analysis
2a.7	"The visualisation could be a very good way of encouraging or enforcing traceability in that once you've got a result, it could give you clear pointers as to where the information has come from, what the previous stages look like and where you could potentially go to readjust that picture."	Tracing visualisation results back to previous stages
2a.8	"The forward direction would be from the system model... we change something in the system model and then we see what's happening in the outcome"	Propagating model changes
2a.9	"The backward traceability, if you want to see or to point something in the analysis results, and see for instance which component contributes to this unpleasant result the most, that's far more non-trivial... our tools will tell you for instance that the system is not sufficient to protect it, and the evidence of that is that the attack game that the attacker will be playing will be more terribly profitable for him. This is basically what the TRE$_S$PASS tools will be able to tell you, but they will not tell you which component in the model is, so to say, the most guilty one, or which component in the model you will need to fix in order to improve the situation. With backward traceability, it also relies on the human judgement and human experience that the human sees what (inaudible) we have or the human can take a look at the intermediate attack tree and see which branches of the attack tree are the ones that give rise to this profitable outcome for the attacker and then see which components or mention in the leaves of the attack tree for instance and then make conclusions regarding which components in the original system model will need to be protected more. So at this level, the traceability will be there but again it involves a lot of human judgement or expert knowledge and in this sense it's not really tool-supported traceability I would say."	Supporting human judgment on tracing critical components from the results
2a.10	"... so especially what was the time... point of time where this analysis was done, and what was the state of the cloud this specific case the cloud that point of time. If I wouldn't have this kind of traceability, yeah that would make my analysis result somewhat more vague, right?"	Tracing back the context of the analysis

Table 5.1 Initial coding frameworks for opportunities of traceability in TRE$_S$PASS (continued)

Code	Interview transcript	Initial coding framework
2a.l1	"And basically, the analysis result should be understandable in terms of the model files in the beginning of the input. So I think we need also traceability that the terms which end up in the analysis results can be uniquely identified back to the data, otherwise we don't really know if you spot a certain risk we wouldn't know to which piece of the infrastructure so to say is it related, and where do we need to fix something or what would be the context to fix something."	Linking analysis results to model components
2a.l2	"...in TRE$_S$PASS it would be nice to be able to know which parts of the TRE$_S$PASS risk model would be affected if something changes in the input data. That, I think is the primary concern."	Propagating input changes
2a.l3	"...through all these stages, you'd like to be able to trace certain value back to the input where it came from, which means traceability from the input data, through the model, through the attack scenarios, to the outputs that are presented to the user. I think in this sense the traceability would follow the stages of what we've called a navigation metaphor. So, the maps, and the routes, and those kinds of things."	Tracking forward input data throughout TRE$_S$PASS process
2a.l4	"you could have traceability for... even which part of the map you could trace that back to the maybe a tag in a picture in the LEGO model, that would be another...we haven't even discussed that, but if you would have a picture of the LEGO model, and somebody would add a node to the formal map, he could basically pinpoint on the page of the LEGO model where that thing is located. That would be again another form of traceability."	Tracing model components back to data
2a.l5	"But quantitative data, if it's some data about social engineering based on a scientific paper, then it could be a link to the scientific paper where it came from. It could all kinds of link... or a link to a vulnerability database, if the classification changes in the vulnerability database, you'd want to update something."	Tracing data back to data sources
2a.l6	"... it would be very nice to be able to link some of these types of models, and to be able for instance to propagate modifications or to propagate interesting results from one model to another. So for instance, if socio-technical model changes, how the generated attack tree changes, this is a big question."	Propagating model changes
2a.l7	"I think to the way I define traceability, it allows you to connect parts in the model with parts of your results. So for example, it would tell you which parts are especially important for risk assessment or which parts have high influence on the outcome of the risk assessment."	Linking analysis results to model components
2a.l8	"...for many of the visualisations it would be very interesting to be able to trace this back to the model."	Tracing visualisation results back to previous stages

Table 5.2: Final coding frameworks for opportunities of traceability in TRE$_S$PASS

Final coding framework	Initial coding framework
Static view (cross-sectional risk assessment)	Tracing back model components to input data Trace back risk calculation Provide support for security decisions Understanding the connection between different TRE$_S$PASS processes Understanding the modelling decisions in the model creation stage Tracing visualisation results back to previous stages Tracing back the context of the analysis Linking analysis results to model components Supporting iterative analysis Tracking forward input data throughout TRE$_S$PASS process Tracing data back to data sources
Dynamic view (longitudinal risk assessment)	Propagating model changes Propagating input changes

To briefly clarify used nomenclatures, *input* and *data* should not be confused with *data sources*. The Common

Vulnerability & Exposures list [1] (CVE), for example, is a *data source* that provides vulnerability *data* of a certain software. An organisation chart document is a *data source* that provides *input* of the organisation structure that is represented in the TRESPASS model.

In the final coding frameworks, we recognised two emerging themes out of the many initial coding frameworks. The first theme belongs to the *static view* of traceability, which acknowledges the function of traceability in a cross-sectional sense of analysis. This view encompasses among others, the functions of tracing back risk calculation and the function of tracing back the context of the risk analysis activity. The second theme falls under the *dynamic view* of traceability, in which traceability is asserted as an enabler to propagate model-related or input changes in risk analysis.

We acknowledged the presence of *surrogacy* during the interviews, in which a stakeholder's viewpoint is represented by other person. More often than not the respondents assumed the role of 'project intermediary' [7] and tried to envisage how traceability in TRESPASS could help target users in achieving certain goals. Although often regarded as a dangerous obstacle [7], we believe surrogacy is necessary evil that needs to be carefully weighed in, in order to formulate plausible expectations and goals of stakeholders that we may not have access to.

To ensure alignment with overall intended impact of TRESPASS, we referred to the official Description of Work document (not publicly available) and derive the trace model's potential contribution to the TRESPASS project. This is conveniently presented in Table 5.3.

Table 5.3: Contribution of trace model to TRESPASS

Seventh Framework Programme impact	**TRESPASS contribution**	**Trace model contribution**
"Improved European industrial competitiveness in markets of trustworthy ICT, by facilitating economic conditions for wide take-up of results; offering clear business opportunities and consumer choice in usable innovative technologies; and increased awareness of the potential and relevance of trustworthy ICT"	**Security of planned ICT systems** *"Organisations developing ICT systems or services themselves may apply the method to the planned system in its intended operational context.* This process will improve the security properties of the ICT systems offered to the public, by early identification and quantification of the information security risks associated with the socio-technical system around the technology to be designed. The method will *increase the transparency of security decisions made*, raise trust in such technologies, and therefore increase market value." "... An organisation using TRESPASS methods and tools will therefore be (1) better prepared for any eventuality, (2) it will be able *to make better judgements about the risks it is willing to take,* and (3) *it will be more capable of justifying the costs of the risks that it wishes to mitigate.*" (emphasis added)	Increased transparency will be supported by the presence of traceability links from analysis results to model components, provisioned by the trace model. The traceability links will be able to show which model components carry the highest risks (e.g. administrator workstation which still runs legacy applications). Security decisions such as countermeasure application (e.g. update to newer OS) can be justified by explicit links showing that indeed the countermeasure will bring highest impact to improve the security measure. These links will be also support risk acceptance and mitigation decisions, namely which risks *not* to mitigate and which risks to mitigate, based on budget or other considerations. In a cloud computing context, this can apply to both cloud service providers (e.g. Microsoft Azure) or high-tech organisations utilising cloud.
"Adequate support to users to make informed decisions on the trustworthiness of ICT. Increased confidence in the use of ICT by EU citizens and businesses. Increased usability and societal acceptance of ICT through understanding of legal and societal consequences."	**Organisational security** "The methods provided will be used by our consultancy partners to analyse and improve information security in organisations offering services to the public, thereby establishing best practices for such risk management, and increasing the trustworthiness of the information handling by these organisations... the method may allow the public to choose between different organisations and different services based on the quantitative risk level calculated by TRESPASS. Governments will be able to choose between different implementation options of a system based on such an analysis." "The main purpose... is for organisations to make well informed decisions on the risk posture of the organisation itself."	Traceability links will enable consultancy companies to easily compare multiple scenarios of countermeasures application to its clients, which is in essence equal to performing sensitivity analysis. This functionality is of course not limited to consultancy companies, but also to organisations who independently perform risk assessment. The capability to perform sensitivity analysis with TRESPASS will indeed lead to more informed security decisions.

[1] https://cve.mitre.org/

Table 5.3 Contribution of trace model to TRESPASS (continued)

Seventh Framework Programme impact	TRESPASS contribution	Trace model contribution
"Demonstrable improvement (i) of the trustworthiness of increasingly large scale heterogeneous networks and systems and (ii) in protecting against and handling of network threats and the reduction of security incidents."	"Application of the method will lead to better prediction of possible attacks, as well as optimisation of investments in countermeasures."	*See first contribution.*

In a more dynamic sense, the trace model will aid propagation of input/data changes or model changes. This will be demonstrated well in our chosen scenario for the cloud case study. Dynamism is in the heart of cloud computing. Demand changes, to mention one instance, can easily happen due to seasonal fluctuation (e.g holiday season sales promotion). Other change drivers include migration of VMs between multiple physical hosts or quick reconfiguration due to disrupted physical infrastructure [190]. The second scenario described in Section 5.2 will reflect examples of the dynamism.

We now discuss the different Slots and Roles in the development of the trace model along with their viewpoints in the following sections.

5.1.1. The System

In this Circle, the **Normal Operator** is the only human stakeholder slot we consider. The **Maintenance Operator** slot and **Operational Support** slot are irrelevant as we do not intend to design an independent tool for the trace model. The Normal Operator slot is highly relevant, as it contains the possible different Roles of end users. As shown in Table 1.1, these end users can vary from an auditor to a Chief Information Security Officer (CISO) of an organisation. However, we believe that the suitable role in a high-tech organisation is an *information security officer*. The Normal Operator role only deals with the 'raw' *output* of the Product (e.g traceability links between analysis results in TRESPASS and the model components involved), and not the processed information. The latter output will be beneficial to (*Functional Beneficiary*) roles to evaluate the security posture of the organisation and consequently support security investment decisions. One of the main requirements of Normal Operators is *operability* [7]. This can be translated to for instance easy interpretation of output. For example, the traceability links should be shown in a form of presentation that enables easy perusal (e.g. plain text information or diagram that shows dependencies between analysis results and data).

5.1.2. The Containing System

In a high-tech organisation, **Functional Beneficiary** roles are security decision makers. These roles could be represented by *risk managers*, or *CISO*. By providing clear links, the trace model can provide clear justifications on for example, why certain risks should be mitigated and why certain risks may be left 'as is'. Typically such decisions would be made by a CIO (Chief Information Officer) or a CISO. These decisions can also be cascaded down into selecting the right countermeasures to be introduced into the organisation (by consulting the links between analysis results and model components), or to check whether certain countermeasures actually improve the organisation's security posture (by running a sensitivity analysis and seeing whether the analysis results are affected).

Roles that are responsible for **Interfacing System** include TRESPASS researchers involved in WP5 (Process Integration) and WP6 (Tool Integration). They will have to ensure seamless fit of the trace model with the overall TRESPASS process and with the rest of TRESPASS tools. Details are presented in Chapter 6, in which we outline the design process by also considering how the trace model can support flawless execution and not hinder the existing TRESPASS process. Tools that potentially should integrate with the trace model in order to produce the desired traceability links are also considered.

Ultimately, the roles of **Purchaser** and **Product Champion** are taken up by members of TRESPASS consortium. These roles are ofttimes crucial in advocating and securing sufficient financial support throughout a project. We will not pay great attention to these roles, as we do not consider financial support as a monumental requirement in developing the trace model.

5.1.3. The Wider Environment

Based on the responses from interviewees, we do not foresee any potential **Negative Stakeholder** that might be harmed with the presence of a trace model in TRESPASS. The general impression we received from different respondents is rather positive, agreeing that traceability can bring added value to TRESPASS. With regards to beneficiary, we believe that shareholders in high-tech organisations can become **Financial Beneficiaries** given the trace model. With enhanced transparency on the TRESPASS process and results, shareholders can make informed choices on appropriate security investments (cf. Table 5.3). The trace model can improve TRESPASS' role as a decision support system in these organisations. We do not see any **Political Beneficiary** of the trace model. There is no direct **Regulator** for the trace model, but those who fill in Regulator roles in TRESPASS might be considered indirect regulators, as the trace model will also be restricted by regulations that bind TRESPASS. The European Commission who supports the majority of the TRESPASS financing and the TRESPASS coordinator are examples of Regulator roles. **Developer** roles are taken up by the student and researchers involved in this research, and **Consultant** roles are assumed by other TRESPASS researchers who contribute to the development of the trace model (e.g. interviewees).

5.1.4. Goal formulation

Based on the final coding frameworks compiled in Table 5.2, we formulate two dimensions of goals G to be fulfilled by the trace model as follows:

1. **Static view goals** (G_1)

   The static view goals are formulated based on the static view coding frameworks shown in Table 5.2, which acknowledges the function of traceability in a cross-sectional/one-time off risk assessment process. These goals are:

   The trace model should enable tracing of data used in the analysis back to the respective data sources;

   The trace model should enable tracing of model components to the (recorded) decisions behind the modelling decisions (e.g. which components are included and which are abstracted out for what reasons; who made what decisions; what the context of the analysis is: when it was conducted, under what conditions and assumptions, etc.);

   The trace model should enable users to pinpoint data throughout the TRESPASS process;

   The trace model should enable tracing of analysis results back to the attack tree calculation, further back to the model components, and finally to see what data are involved; and

   The trace model should support accountability in the risk assessment process and enable decision makers to defend security decisions made based on the risk assessment results.

2. **Dynamic view goals** (G_2)

   The dynamic goals are formulated based on the roles of traceability in a longitudinal view of risk assessment, which includes:

   The trace model should be able to support propagation of changes in the input data/model components throughout the analysis process. The analysis results should reflect the changes made.

The traceability from visualisation results to previous stages is given less priority at the moment, as visualisation processes are still under active development and are subject to change.

Clearness

A large number of the formulated goals have been derived directly from direct speeches of the interviewees. The interviewees have been clear in stating their preferences, which are mostly supported with examples and additional explanation.

Stakeholders' acceptance

Essentially, the goals are derived from the stakeholders' wishes (including those who are indirectly involved through surrogacy), hence it would logically follow that these goals are accepted by the stakeholders.

Feasibility

The biggest constraint in developing the trace model is the time constraint of TRESPASS project, which will be finalised by October 2016. In order to meet this time constraint, the researchers have explicitly stated that it is best that the trace model will not be constructed as a tool, but rather as a set of frameworks with guiding principles.

Affordability
The research is designed to be carried out without incurring additional cost from what has been budgeted in the TREsPASS project. We do not foresee any unexpected overhead cost.

Opportunity and opportunity cost
An obvious opportunity cost that could fulfil (parts of) this research's objective is to extend the existing tools to embed capabilities of traceability. However we were informed that another research in parallel has attempted to program an extension of the Tree Maker analysis tool. We believe our research can complement that work by pleading the needs for traceability in a more high-level view in cyber risk assessment, which is technically demonstrated by the other research.

Order of priority
Our decision in ordering priorities for developing the trace model moves according to the level of difficulty. We first prioritise the static view of traceability, as it is inherently present in the process as it stands now. We move one level up by deploying traceability in its dynamic sense, namely by taking care of changes. The lowest priority would be given to development of traceability from the visualisation stage back to the previous stages, as the visualisation stage is still under active development and is under constant redefinition.

5.2. Identifying the phenomena, their causes and effects and their contributions to the stakeholders' goals
In this section, we elaborate the phenomena relevant to the problem and recognise their causes and effects. This boils down to devising possible (attack) scenarios. The scenarios are derived to illustrate possible phenomena that can happen in the cloud and to make the process of designing the trace model less abstract.

5.2.1. The cloud case study
The cloud case study we use in this thesis is derived from the TREsPASS project, which was used in the 2015 TREsPASS Social Engineering Challenge [2]. It involves the fictional company *International Traders Ltd (ITL)*. Details of the case study is presented in an 'as is' format from the flyer as follows:

International Trading Ltd. (ITL)
Imagine the medium sized company called International Traders Ltd. (ITL). ITL is a commodity trader – they buy and sell oil and gas related investments around the world for an international set of clients. They own an office with a co-located datacenter.

*Much of their trading is automated and they have a large IT department. The key company asset **fileX** is a list of international clients with the following details:*

1. *The clients names and address*
2. *Bank Account details*
3. *Clients Investment portfolio*
4. *Details of investment transactions dating back 5 years*

ITL has a number of departments including:

- *A **marketing and sales** department that is responsible for developing investment packages*
  The marketing data is informational in nature and not considered sensitive
  The marketing and sales data is stored with all other non sensitive data on the marketing computer system
- *A **trading** department that handles all transaction with clients*
  The client data and financial transactions are considered sensitive information and are stored on a separate trading computer system
- *An **IT** department that is responsible for the companies computing infrastructure. This includes:*
  Networks, switches, routers etc.
  Servers
  Applications

Some key people in the company are:

[2] http://trespass-project.eu/sites/default/files/Deliverables/TREsPASS_SEAward2015_0.pdf

Table 5.4: *Actors in the cloud case study*

Actor	*Job/Role*	*Department*	*Physical Access*	*Logical Access*
Ethan	*Fraud investigator*	*Trading department*	*Up to Open Office*	*To VM1*
Finn	*Finance manager*	*Marketing department*	*Up to Open Office*	*To VM3*
Terry	*Technician*	*IT department*	*Everywhere*	*No access*
Sydney	*IT system administrator*	*IT department*	*Everywhere*	*Full access*
Cleo	*Cleaning personnel*	*External contractor*	*Everywhere*	*No access*
Grey	*External visitor*	*N/A*	*No access*	*No access*
Big	*CEO*	*N/A*	*Up to Open Office*	*Read access*

Their main office building is located in an engaged industrial area and consists mostly of an open office space and the data centre room, shown in Figure 5.2.

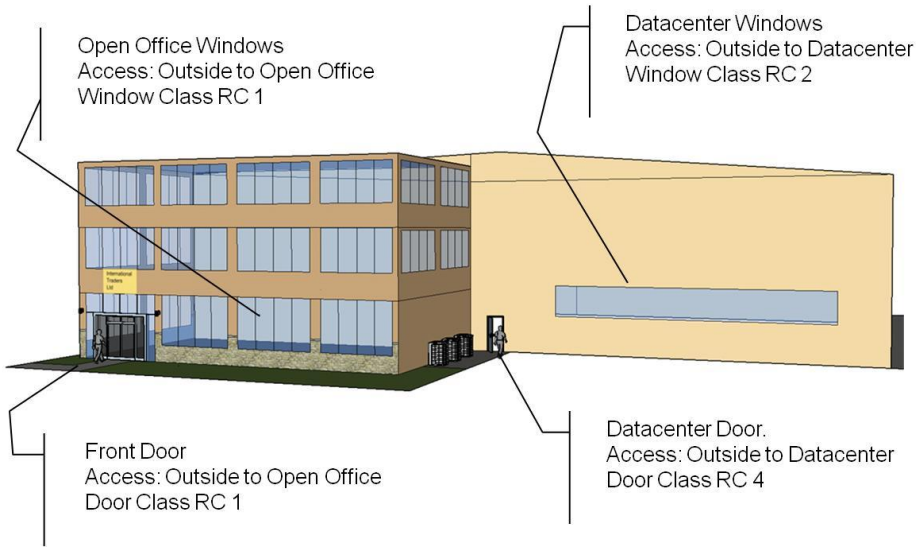


Figure 5.2: *ITL office (exterior view)*

Source: [187]

The office is open plan. On the ground floor is front door access to the office space. All employees have a badge access through the front door [Security Class RC2].

An internal door leads directly into the data center. This door is access controlled and only data center staff has access [Security Class RC1].

The bottom floor of the open office has a number of large windows [Security Class RC1].

The data center is directly attached to the office space from where it is accessible through an internal door. This door is access controlled and only data center staff has access [Security Class RC1].

There is a separate external door directly into the data center. This door is used to bring in computer supplies. It is a secure door that can only be opened from the inside [Security Class RC4].

The data center has two large windows [Security Class RC2].

The cloud infrastructure is running on two physical servers, both located in the Data Center. On each server, two virtual machines, a virtual switch and a virtual firewall are running on top of a Hypervisor. These virtual components are connected to physical network components, namely switch SW2 and a Gateway. Through this connection it is possible to reach the physical and virtual infrastructure from the Laptop.

*The sensitive document **fileX** is located in the storage of VM1. This is currently stored together with the trading solution. The marketing department has a separate solution. Both systems are on physically separate host systems within the data centre.*

To provide a more realistic scenario, doors and windows are classified according to their security class as follows:

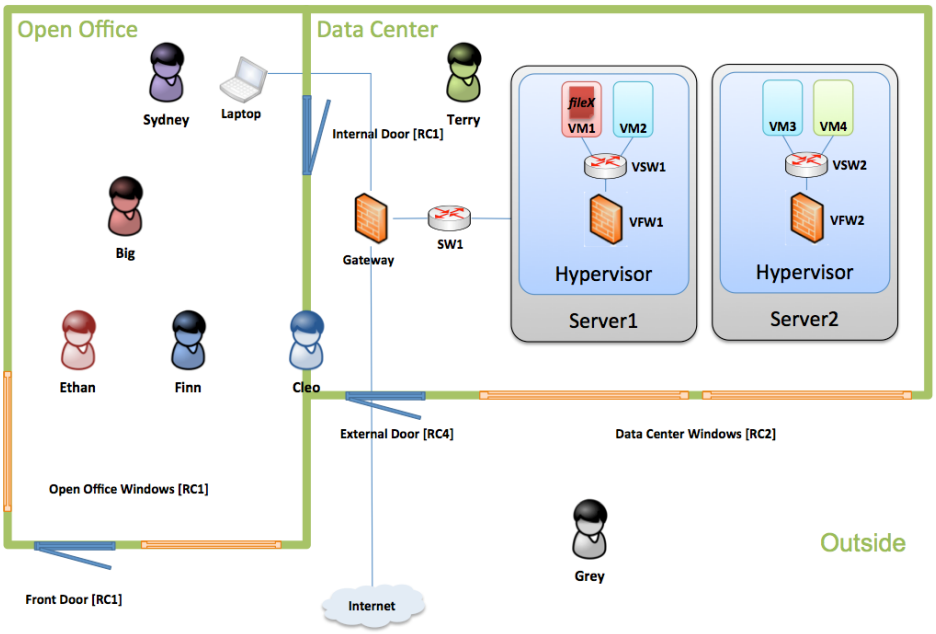


Figure 5.3: *Office plan setup*

Source: [187]

Table 5.5: *Doors and windows security classifications*

Security class RC	*Burglary type*	*Use*
1	*Basic protection against burglary by the use of physical force*	*Premises without increased risk of burglary*
2	*Occasional burglar uses basic burglary tools*	*Increased protection for normal housing security*
3	*Experienced burglar uses heavy duty drilling and hammering tools*	*High level of security for the premises in view of increased risk of burglary*
4	*Experienced burglar additionally uses electric saws and power drilling tools*	*Extra high level of housing protection in view of high risk of burglary*
5	*Experienced burglar additionally uses power cutting and heavy duty drilling tools*	*Extremely high protection for military and industrial premises. Minimum door weight- 300 kg*
6	*Experienced burglar additionally uses power cutting and heavy duty drilling tools*	*Extremely high protection for military bunkers, etc. Minimum door weight- 500 kg*

We now proceed with describing the scenarios that complement the cloud case study. The first scenario is used to describe the *static view* of traceability, and the second scenario the *dynamic view*. In order to be able to successfully represent an attack scenario that spans the physical, digital, and social domains of a cloud computing environment, we choose a social engineering attack scenario for the former, and a scenario involving various plausible socio-technical changes for the latter. This distinction will be made crystal clear in Chapter 6 with necessary explanation, but in this chapter we limit ourselves to only describing the scenarios. We demonstrate these scenarios using Use Cases applied to the cloud case study.

5.2.2. Scenario 1: Social engineering attack

According to Hadnagy, social engineering is "the act of manipulating a person to take an action that may or may not be in the 'target's' best interest." [72]. The chosen scenario is the winning scenario of the 2015 Social Engineering Award written by David Kelm of IT Seal [3]. It consists of several underlying assumptions and two steps needed to launch the attack. Similar to the cloud case study, the scenario is presented as follows in its 'as is' format. Other strong motivations behind the selection of this scenario for this thesis are its feasibility and 'realisability', which was acknowledged by the panel of judges composed of experts in cyber security.

[3] `http://trespass-project.eu/sites/default/files/Deliverables/SecEngAward2015-Winner.pdf`

Assumptions

- *ITL employees are assumed to know one another and are able to communicate easily.*
- *None of the involved persons is liable to bribery.*
- *ITL has a safe technical environment. Deploying a Trojan file or a similar malicious file to obtain the target file is not possible.*
- *There is only one Laptop to access the VMs and one is able to access **fileX** just in combination with the correct login credentials of Big, Ethan or Sydney (to access VMI). The badges authorise the owner to enter the respective area without restrictions.*
- *Ethan and Sydney are aware of frauds (as fraud investigator and IT system administrator).*
- *Big as CEO is less aware and more customer-oriented. Because he is concentrating on running the business, Big is susceptible to a social engineering attack.*
- *The attacker(s) has only one shot. Once access to Laptop is gained, passwords are harvested from Big in multiple attempts.*

*The attack consists of two parts: **Obtaining correct login credentials** and **getting physical access to the laptop** (since it is in general considered safe).*

Part I: Gathering login credentials

To obtain Big's login credentials there are several options:

1. *Call Big and pretend to organise an **Investment-Conference** (creating a background story, setting up a webpage) and have him considered as a keynote speaker. Big would be one of the main candidates, which provides reasons to fish for his CV to be placed on the web and on the flyer. To do so, Big needs to register himself on the conference webpage and uploads the file (led to webpage during call). (GOT EMAIL, GOT PASSWORD1, GOT CV). Together with the CV information and crawling from social media sites, the attacker can attempt a **personalised password guessing** (brute-force) attack and try to break into an e-mail or other accounts (may be reused for business purpose). (GOT PASSWORD2)*
2. *In case the attacker can figure out what the **intranet** looks like, the login page might be **cloned**. When Big is on travel, the attacker can send a spoofed E-Mail in Sydney's name and ask to check some critical issue with a customer. To do so he needs to login with his company-account on the cloned webpage. After submitting he is redirected to the real intranet. (GOT PASSWORD3)*
3. *Offer to install a **video surveillance** component in the open office (creating a background story, setting up a webpage), at a competitive price. The attacker can then use our backdoor access to observe the employees (not just Big) using the Laptop and entering their login data - a technical form of **shoulder surfing**. (GOT PASSWORD4)*

Several of these options can be executed since they are all designed not to be identified as attacks.

Part II: Gaining Laptop access

*To gain physical access to the laptop, the attacker makes use of the business relationship with the **external contractor** (EC). Shortly after obtaining the login credentials (credentials could be changed, if waited too long), EC is called in the afternoon by spoofing Big. The attacker claims not to need Cleo this week (due to some renovation of the Open Office) and asks EC to inform Cleo directly. Moreover, the attacker gives an alternative phone number for callbacks, since Big is travelling to a conference this week.*

*Secondly, the attacker calls Big by spoofing EC and claim that Cleo is sick for about a week. To satisfy customer, the attacker promises to send a replacement: Henry. Since he has no badge to access the area and Cleo's badge is not available, Big is asked whether the IT department could grant a temporary badge to be able to enter the building freely. Once Henry (background story and costume are prepared) gets his badge he can enter the building, particularly the open office. To access the Laptop, Henry enters the building at night. In case the badge just works in the office hours, Henry comes in late and stays longer, until everybody has left. Thus, Henry can have access to the Laptop, login and access **fileX** when the room is empty.*

5.2.3. Scenario 2: Changes

The cloud environment can be very dynamic, with changes happening in its physical infrastructure (servers, switches, etc.), virtual infrastructure (virtual machines), and its socio-technical space (the employees of cloud providers and their access control) [190]. Through a more generic lens, changes that bring impact to organisations may concern the following items [195]:

- The enterprise architecture;
- Known vulnerabilities;
- Asset values;
- Attacker profiles;
- Attacks or attack attempts occurring in the organisation;
- Attacks or attack attempts occurring in similar organisations;
- Organisational culture; and
- Standard map components used in the model.

In this scenario, we intend to show changes that might bring notable impact to the organisation. Due to the limited access to the TRE$_S$PASS tools for the time being, we select changes that are not cloud-specific. The first of these is changes in *known vulnerabilities*. This is chosen because it concerns the external environment and is very likely to happen in real world as well. Online vulnerability databases such as CVE and NVD (National Vulnerability Database)[4] are being updated on a daily basis, clearly showing that more and more vulnerabilities are being discovered. For this type of change, we focus on new vulnerabilities discovered for Windows 10, as most of the managers recently had an upgrade to Windows 10.

To represent internal changes, we choose to take into account changes in *asset values*. The first is exemplified by the fact that ITL has recently secured a series of deals with fantastic figures in the last two months. Coincidentally, these transactions happen after the organisation performed its most recent risk analysis. Of course, these transactions get recorded in ***fileX***. According to Finn's estimation, the value of these recent transactions inflates the value of ***fileX*** by roughly 20%. Another internal change is brought about by the arrival of Liam, a new IT system administrator that works alongside Sydney. This is ITL's attempt to exercise the four-eyes principle. Liam is a fresh hire who is in still in training and is still working his way around to understand ITL's IT infrastructure. However, he already has experience working for two years as an administrator in a small university. He was hired as a backup for Sydney, remembering that ITL's IT infrastructure has grown into a complexity that is too difficult to be managed by one person.

ITL's policy requires a risk assessment activity to recur every six months. However, important changes have been taking place only two months after the last TRE$_S$PASS-assisted risk assessment. These important changes might be too crucial to neglect, and therefore ITL plans to conduct yet another risk assessment activity now. The plan is not to conduct a full-scale assessment from scratch, but rather to work incrementally from the results of the previous assessment.

5.3. Ex-post evaluation of problem investigation phase

We end this chapter by performing checks on whether the problem investigation phase has been correctly carried out, using a set of questions proposed by Verschuren and Hartog [213]. The first question is *"Was the involvement and variety of experts well balanced against the expected impact of the design?"*. This question probes whether there is a relation between the design effort and its expected impact. We believe such a relation exists. Experts with various backgrounds have been involved in the data collection stage (the semi-structured interviews). The experts did not only express their own views and wishes, but also those of the end users of TRE$_S$PASS. Out of the nine experts, five are currently engaged in academia and four are employed in various industries (e.g. research, consultancy). This mix of experts have been carefully selected, and the interview results reflect that these experts view the traceability design problem from different angles.

The second sanity check is *"Is G sufficiently sharp defined in order to derive the functional requirements (R_f), user requirements (R_u), and contextual requirements (R_c) and to give direction to the next stages in the designing process?"*. We believe this is the case, as already demonstrated by formulation of two distinct classes of goals in Table 5.2. Further requirements are devised later in Chapter 6, but the goals have been clearly formulated in order to easily derive the requirements and assumptions.

Lastly, we reflect on the question *"Have standard methodological guidelines for empirical research been followed during the process that led to the formulation of G?"*. We indeed conducted the interviews following common methodological guidelines. Limitations have also been clearly explicated in Chapter 4. *Validity* of the interview questions have been checked by the research supervisor to avoid mistakes such as *leading questions* or *double-barrelled questions*. *Reliability* of the interview data processing is established by following a certain methodology that is fashioned after grounded theory [31]. *Researcher-independence* is ensured without any vested interests shadowing the interviews, and finally *verifiability* is achieved by providing the transcripts both to the interviewees and the graduation committee of the thesis.

5.4. Summary of problem investigation phase

This phase has resulted in important items that serve as building blocks for the design of the trace model, namely:

- Identification of important stakeholders, including operators and beneficiaries of the trace model;
- Formulation of static view goals (G_1) and dynamic view goals (G_2); and
- Formulation of scenarios that represent both types of goals.

[4] `http://nvd.nist.gov/`

6

Treatment design

This chapter is a continuation of the TRESPASS-specific discussion we began in the previous chapter and largely concerns the second step of the design science methodology used to develop the trace model. In this chapter, we explicate the process of designing the trace model, starting from how we come up with the requirements and how they become involved in the design process of the trace model. There are three main steps in designing the trace model: requirements and assumptions specification, available treatment identification, and designing the trace model itself. We begin with showing the process of eliciting different types of requirements that the trace model has to satisfy and continue with defining criteria based on these requirements. We finally decompose the criteria into less abstract specifications. At the end of this chapter, an evaluation encompassing the requirements, criteria, and specifications is explicated.

6.1. Specifying the requirements and context assumptions

As pointed out in Chapter 3, requirements are properties that the treatment/artefact should possess, which in this case is the trace model. We elicit requirements from different sources, namely:

1. **Goals**

   The desired goals for traceability have been outlined in Chapter 5 (Table 5.2). We further break down these goals into requirements to be fulfilled by the trace model. By nature, these requirements would mostly be functional requirements (R_f), as they are inferred from functional goals.

2. **Interview results**

   In the semi-structured interview, the second question regarding traceability in TRESPASS (cf. Appendix C) specifically inquires the interviewees regarding requirements that they would like to see in order for traceability to be seamlessly supported in TRESPASS. We use the answers to these questions to complement the requirements that are deduced from the goals in the first point.

3. **Use Cases**

   Use Cases will help us in formulating requirements based on possible interaction between the user and the TRESPASS tools.

Each source of requirements will be individually analysed in the subsequent sections.

6.1.1. Goals and interview results

We first begin by revisiting the goals from Section 5.1.4, wherein the static view goals (G_1) and dynamic view goals (G_2) of the trace model are presented. These goals are given as follows:

1. **Static view goals (G_1)**

   The trace model should enable tracing of data used in the analysis back to the respective data sources ($G_{1.1}$);

The trace model should enable tracing of model components to the (recorded) decisions behind the modelling decisions (e.g. which components are included and which are abstracted out for what reasons; who made what decisions; what the context of the analysis is: when it was conducted, under what conditions and assumptions, etc.) ($G_{1.2}$);

The trace model should enable users to pinpoint data throughout the TRESPASS process ($G_{1.3}$);

The trace model should enable tracing of analysis results back to the attack tree calculation, further back to the model components, and finally to see what data are involved ($G_{1.4}$); and

The trace model should support accountability in the risk assessment process and enable decision makers to defend security decisions made based on the risk assessment results ($G_{1.5}$).

2. **Dynamic view goal** (G_2)

   The trace model should be able to support propagation of changes in the input data/model components throughout the analysis process. The analysis results should reflect the changes made ($G_{2.1}$).

We now decompose these goals into requirements. At this stage, the process is done by first looking for important elements enveloped in these goal statements (e.g. 'data', 'model components', 'analysis results') and subsequently consulting documents (especially deliverables) that might contain additional information on these terms. Information from these documents aids us in delivering more concrete requirements. Where necessary, references to these documents are appended in footnotes.

1. **The trace model should enable tracing of data used in the analysis back to the respective data sources**[1]

   The trace model should be compatible with different types of data used in TRESPASS (e.g. physical data, social data, commercial data, etc.);

   The trace model should be able to discriminate different types of data used in TRESPASS (e.g. vulnerability data, stakeholder goals, technical data, etc.);

   The trace model should be able to discriminate multiple sources of data (e.g. data retrieved from data collection stage; data retrieved from databases such as vulnerabilities or scenarios, etc.); and

   The trace model should be integrated with the knowledge base as the "central data repository" and the TRESPASS model.

2. **The trace model should enable tracing of model components to the (recorded) decisions behind the modelling decisions (e.g. which components are included and which are abstracted out for what reasons; who made what decisions; what the context of the analysis is: when it was conducted, under what conditions and assumptions, etc.**[2]

   The trace model should be able to incorporate different types of modelling tools in the modelling process (e.g. ArgueSecure, LEGO model, manual drawing + meeting notes, etc.);

   The trace model should enable annotation in the manual modelling process leading up to the final model (e.g. place tags in drawings, insert actors in meeting notes, etc.); and

   The trace model should be integrated with the different libraries used in the model creation process (e.g. Model Pattern Library Attacker Profile Library, etc.).

3. **The trace model should enable users to pinpoint data throughout the TRESPASS process**

   The trace model should be able to uniquely identify data used in the risk assessment process.

4. **The trace model should enable tracing of analysis results back to the attack tree calculation, further back to the model components, and finally to see what data are involved**

   The trace model should be integrated with the whole TRESPASS process (from the data collection stage up to the visualisation stage); and

   The trace model should be able to store intermediate parameters generated during the risk assessment process that are not currently stored in any of the TRESPASS tools.

[1] D2.1.2 [191] lists all data sources and required data types in TRESPASS.
[2] D5.3.2 [194] lists different model creation strategies available in TRESPASS.

5. **The trace model should support accountability in the risk assessment process and enable decision makers to defend security decisions made based on the risk assessment results**

   The trace model should enable explicit reasoning behind security decisions taken and risk assessment results; and

   The trace model should be capable of storing the arguments behind security decisions to enable future reference.

6. **The trace model should be able to support propagation of changes in the input data/model components throughout the analysis process. The analysis results should reflect the changes made**

   The trace model should be able to identify changes made in input/model components in an analysis as compared to a prior analysis;

   The trace model should be able to store different analysis results to enable comparison; and

   The trace model should be able to show the differences between multiple analyses.

After the decomposition of goals into requirements from relatively passive sources, we once again take a look at the results of the interview pertaining the wishes of the researchers regarding the traceability support to dig empirical insights and identify requirements, as shown in Table 6.1.

Table 6.1: Requirements based on interview results

Code	Interview transcript	Identified requirements
	Interviewer: "What do you think are the necessary requirements to support traceability in TRE$_S$PASS? What features should a traceability support for TRE$_S$PASS have?"	
2b.1	"... one of the things that I think in TRE$_S$PASS is very important is to be able to trace back what the levels of aggregation of different datasets are."	Trace back data aggregation level
2b.2	"Other issues that we should be able to trace back is how the recent the datasets are."	Trace back how recent the datasets are
2b.3	"...another useful requirement is being able to do a sensitivity analysis or something like what you say, if I change something in the model well how does that influence my result? This is already sort of possible because you can always change the model and rerun the analysis, but it would be nice if the tool could maintain a sort of log or history of these changes, right?" "A log or an archive of previous analyses would also be useful for traceability reasons, I think."	A log of previous analyses
2b.4	"...for me the most crucial thing is to be able to take the results of the analysis and get back to the model and I think even that is quite difficult, so for me that's the bottom line."	Linking analysis results to the model
2b.5	"So to go back from the analysis results to the TRE$_S$PASS model, which means you would need all the intermediate tools to support this, right? You'll need the Attack Tree Analyzer to support for this information. You would most notably need the Attack Tree Generation by the Tree Maker to support this. Attack tree augmenting and annotation is not that critical, they're pretty small tools. But yes, the Attack Tree generation is the most notable component that would need to support this sort of going backwards. It should somehow record the back links because the Attack Tree Generation sees the system model and also the attack tree, so it somehow should annotate the leaf of the attack tree saying that this leaf is here because the model has a component xyz. So the most critical component in this context would be the Tree Maker."	Integration with the TRE$_S$PASS tools
2b.6	"the visible central component, when he creates a new model, he creates an identifier, and with that identifier all information will be stored in the knowledge base. . . .with this kind of versioning, we would have the opportunity to see what's going on and keep track of the different states of the system, so that you can also afterwards see what you changed and so on....all required information at that point and state is kept at that point and state in the context of this model ID."	Generation of unique identifier for data
2b.7	"I think we would need an additional database to store data from different steps that we are currently not doing...we need to store all this data or intermediate parameters that we are not storing at the moment. It can be probability of attack, it can be risk, it can be cost, it can be attack time."	Storage for intermediate parameters

Table 6.1 Requirements based on interview results (continued)

Code	Interview transcript	Identified requirements
2b.8	"Of course it needs to be aligned with the different stages that we have in TRE$_S$PASS which I already mentioned. I think that is the most important... of course it also needs to be aligned with the TRE$_S$PASS concepts, so the actors, the locations, the policies."	Alignment with TRE$_S$PASS concepts and processes
2b.9	"I think the really necessary thing that we need is the link between the socio-technical model and the attack trees, or attack models."	Link between TRE$_S$PASS model and attack trees
2b.10	"... in theory you can have many links, so for instance attacker profile and the TRE$_S$PASS model, because the attacker for us is also an actor in the model, so to be able to kind of precisely identify which is this actor or vice versa we have an actor how do we create an attacker profile for him, knowing that he is for instance, an employee in this organisation."	Link between TRE$_S$PASS model and attacker profile

Based on the interview results, we come across explicitly stated requirements in the form of wishes conveyed by the interviewees. These requirements can be mapped against the goals we identified earlier. For example, the requirements identified in item 2b.1 (*Trace back data aggregation level*) and 2b.2 (*Trace back how recent the datasets are*) support the fulfilment of $G_{l.1}$ (*The trace model should enable tracing of data used in the analysis back to the respective data sources*). The requirement to keep a log of former analyses (2b.3) helps satisfy $G_{l.2}$ (*The trace model should be able to support propagation of changes in the input data/model components throughout the analysis process. The analysis results should reflect the changes made.*). Requirements 2b.6 (*Generation of unique identifier for data*) and 2b.7 (*Storage for intermediate parameters*) contribute to $G_{l.3}$ (*The trace model should enable users to pinpoint data throughout the TRE$_S$PASS process*). Finally, requirements 2b.4 (*Linking analysis results to the model*), 2b.5 (*Alignment with TRE$_S$PASS concepts and processes*), 2b.8 (*Integration with the TRE$_S$PASS tools*), 2b.9 (*Link between TRE$_S$PASS model and attack trees*), 2b.10 (*Link between TRE$_S$PASS model and attacker profile*) collectively lend themselves to $G_{l.4}$ (*The trace model should enable tracing of analysis results back to the attack tree calculation, further back to the model components, and finally to see what data are involved*).

Moreover, we also asked the interviewees to show the trace links they would like to see being established in the TRE$_S$PASS process with the aid of a superimposed integration diagram. The interviewees indicated that some links are more important than others and that priority should be given to these links during the development of the trace model. These links include:

- Link between *visualise results* process (with Attack Navigator Map/ANM) and *visualise attack tree* process involving the ADTool;
- Link between *visualise attack tree* process with ADTool and the TRE$_S$PASS model;
- Link between analysis results and *visualise attack tree* process with ADTool;
- Link between the analysis results and the *generate attack trees* process with Tree Maker;
- Link between the analysis results and the annotated attack trees;
- Link between *analyse attack tree* process done with ATtack Tree Evaluator, Attack Tree Analyzer, and ATCalc and the knowledge base;
- Link between the annotated attack trees and the Attack Pattern Library;
- Link between the annotated attack trees and the knowledge base;
- Link between the annotated attack trees and the generic attack trees;
- Link between the annotated attack trees and the TRE$_S$PASS model;
- Link between the *augment and annotate* process involving the Attack Pattern Library Tool (APLTool) and the knowledge base;
- Link between *generate attack trees* process with Tree Maker and the TRE$_S$PASS model;
- Link between the knowledge base and the Attacker Profile Library;

- Link between the knowledge base and the Entity Type Library;
- Link between the knowledge base and online vulnerability databases;
- Link between the knowledge base and the binary technical data;
- Link between the generic attack trees and the TRESPASS model;
- Link between the TRESPASS model and the Model Pattern Library;
- Link between the TRESPASS model and the Attacker Profile Library; and
- Link between the TRESPASS model to the LEGO model or other modelling tools used in the data collection stage.

The desired trace links based on the interview are presented in Figure 6.1 as follows.

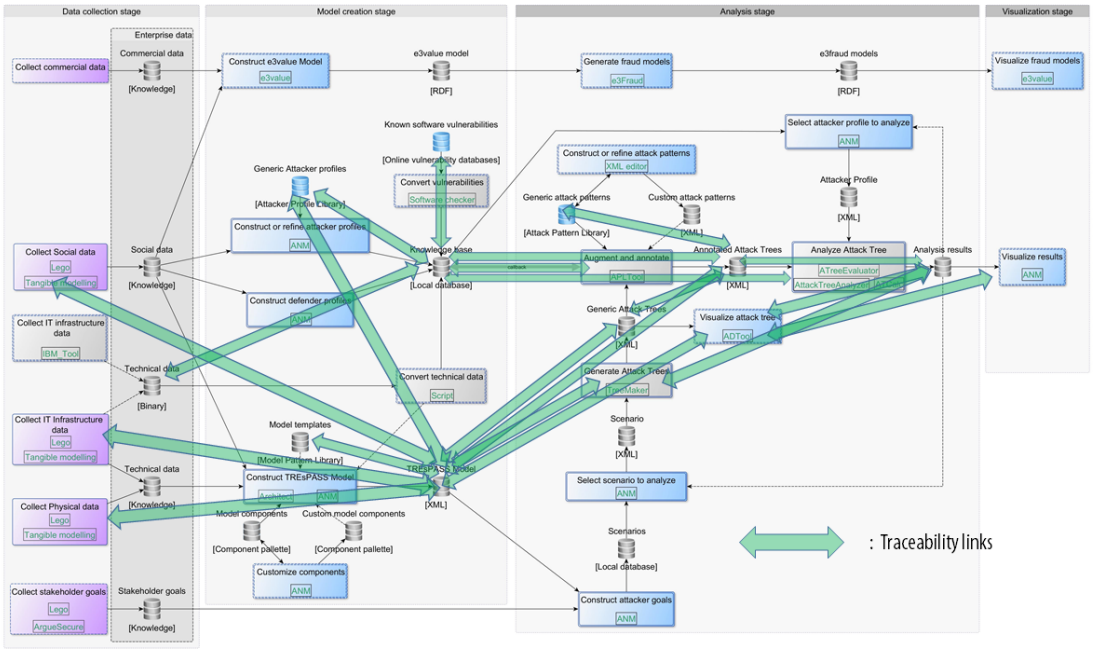


Figure 6.1: Desired trace links

Note: Integration diagram is adopted from [190], green arrows are added.

Onwards, we shall take into account these suggested links into the design process instead of the exhaustive links we provided during the interview (shown in Appendix C). Next, plausible Use Cases are provided to devise more requirements based on possible user interaction with the TRESPASS tools.

6.1.2. Use Cases

Use Cases are devised to understand how the goals can satisfy the wishes of the stakeholders through interaction with the system. The Use Cases can also be used to formulate credible assumptions regarding the functions and the future users. These Use Cases are formulated both from documents on TRESPASS functions and from the interview results. With Use Cases, all required system behaviours can be defined, and essential functionalities can be derived [195]. Use Cases for the *complete* TRESPASS toolkit are documented in D6.2.2. Functions (F) of TRESPASS are also described in the same deliverable, which are further decomposed into Services (S). A number of these Services can be supported by traceability, namely:

- Impact assessment (S1.5);
- Data extraction (S2.1);
- Risk analysis (S3.3);

- Root cause analysis (S3.4);
- Sensitivity analysis (S3.5); and
- Dynamic (re-) analysis (S3.7)

In this section, we limit our scope to Use Cases for traceability support. From the aforementioned functions, existing Use Cases in D6.2.2, and the interview results, four possible Use Cases identified for traceability are identified, namely:

1. **Use Case 1: Understanding Analysis Results**

   This Use Case presents a possible scenario in which traceability can act as a support for linking analysis results to model component and provide explicit justification for current state risk.

2. **Use Case 2: Security Investment**

   In this Use Case, traceability is regarded as a support to justify security investment decisions based on analysis results.

3. **Use Case 3: Change Impact Analysis**

   For this Use Case, we frame traceability as a support to analyse the impact of propagated changes in input data (e.g. data sources) or system changes (represented in the model).

4. **Use Case 4: Sensitivity Analysis**

   Lastly, we introduce traceability as a support to perform sensitivity analysis, namely to see how much influence a certain change in the input brings to the analysis result.

More comprehensive explanation on each Use Case is given as follows. Functional goals or needs and the workflow for each Use Case is described, as well as the required functionalities[3]. Most of the processes in the workflow have already been described in D6.2.2. Additional processes will be printed in **bold** typeface. For all Use Cases, we assume the same type of regular user.

Use Case 1: Understanding Analysis Results

Functional goals/needs

The user has run a risk assessment using TREsPASS tools, but the user wants the analysis results to be described in terms of the model components. Which assets are the 'weakest links' [140]? Which assets are most likely to be compromised? Which controls are least effective and might need replacement?

Description (workflow)

1. The user opens the interface, which shows an empty map of the organisation. The user inserts several standard elements onto the map and draws the relevant connections. If an Archimate model already exists, this may be used as a basis instead.
2. The user clicks on the items to select the corresponding properties, which may consist of asset values, potential damage upon failure, access control mechanisms and an estimation of their strength. Default values can be used if information is unavailable.
3. The user answers a number of general questions about the organisation in survey style (e.g. organisational culture). The answers are used to set properties of the items of the map, in particular the people/employees.
4. **The user inserts additional information regarding the modelling decisions in plaintext format (in submenu 'Comments' for a more generic description, and in each model component for specific explanation). This information could be the context of the organisation when the model was created, the purpose of the analysis, important assumptions, components exclusion and inclusion decision, etc.**
5. The user answers a number of general questions about the threat environment. Based on the answers, the system suggests relevant attacker profiles. The user can select the attacker profiles of interest, and adapt these if necessary.

[3] additional to the ones already described in D6.2.2. Existing functionalities will be indicated.

6. The user starts the analysis. A prioritised list of scenarios/attack trees is presented.

7. **The user selects the scenarios/attack trees to understand which components are linked to a specific scenario/attack tree. Upon clicking on a certain leaf node, the model shows model components that are involved, such as assets (annotated with their quantitative values), control mechanisms, and actors.**

8. The model can be saved for future adjustments.

Required functionalities

- Creation of an empty model [**U1.3**]
- Conversion of model from ArchiMate [**U1.4**]
- Drag and drop of model elements (from library) [**U1.5**]
- Properties menu associated with model elements/entities (available upon click) [**U1.6**]
- **Support for additional submenu to insert plaintext input during model creation (for Step 4)** [**TR1**]
- Survey mode or menu for general project properties (e.g. organisational culture) and attacker properties [**U1.8**]
- Selection of attacker profiles (based on survey) [**U1.9**]
- Menu for adaptation of attacker profiles [**U1.10**]
- Ranking of attack scenarios [**U1.11**]
- **Support for generating unique identifier for every model elements (for Step 7)** [**TR2**]
- **Linking of model components as unique entities in different processes (for Step 7)** [**TR3**]
- Functionality to export and/or save model [**U2.15**]

Use Case 2: Security Investment

Functional goals/needs

The user uses the TREsPASS suite to decide on investments in information security. The user wants to finds out which countermeasures are most cost-effective and therefore decide on the most suitable investment. Another possible goal is to measure Return on Security Investment (ROSI) [33] by using TREsPASS suite to calculate the effectiveness of certain countermeasures.

Description (workflow)

1. The user opens the interface, which shows an empty map of the organisation. The users inserts several standard elements onto the map and draws the relevant connections.

2. The user clicks on the items to select the corresponding properties, which may consist of asset values, potential damage upon failure, access control mechanisms and an estimation of their strength. Default values can be used if information is unavailable.

3. **The user inserts additional information regarding the model components in plaintext format (submenu 'Comments' or something similar). This information could be the context of the organisation when the model was created, the purpose of the analysis, important assumptions, components exclusion and inclusion decision, etc.**

4. The user answers a number of general questions about the organisation in survey style (e.g. organisational culture). The answers are used to set properties of the items of the map, in particular the people/employees.

5. The user answers a number of general questions about the threat environment. Based on the answers, the system suggests relevant attacker profiles. The user can select the attacker profiles of interest, and adapt these if necessary.

6. The user starts the analysis and selects one or more of the following results:

   A list of possible attack scenarios, ordered by their risk;

   A list of possible countermeasures, ordered by their cost-effectiveness; or

   A visualisation of the map, showing 'the weakest links', for example by increasing their size, or changing their colour.

7. **The user annotates explanations of security decisions next to the analysis results in plaintext format. For example, the user can type in one or more security decisions that will be made in the next three months (or any time period) which corresponds to preventing a specific attack step/scenario. This explicitly shows couplings of decisions and the danger to be mitigated. Alternatively, the user can also provide a general list of possible countermeasures, among which the user selects a few to be applied within the next six months (or any time period).**

8. The model can be saved for future references or adjustments.

Required functionalities

- Suggestion for countermeasures [**U2.10**]
- Ranking of countermeasures sorted by RO(S)I [**U2.11**]
- Navigator map visualisation (that can reflect analysis results) [**U2.13**]
- **Support for flagging and manual annotation in the analysis results in plaintext format (for Step 7)** [**TR4**]

Use Case 3: Change Impact Analysis

Functional goals/needs

The user works in an organisation with a policy to run periodical risk assessments. The user has run a risk assessment with TRESPASS tools in the previous period. Although the next cycle is yet to come, the user observes substantial changes in many input sources, both internally and externally (e.g. new network configuration, or new emerging threats). The user now wants to run another analysis before the next cycle arrives.

Description (workflow)

1. **The user receives push notifications from one of the available data extraction tools or requests automated data extraction[4]. The push notifications inform the user that there have been updates in the data based on changes detected by the input tools (see Step 3).**[5]

2. The user opens the file containing the model used in the previous assessment. The file also contains the previous analysis results.

3. Following up on the push notifications, the user adapts the model where needed to represent the changes observed, by moving, deleting, or adding elements and changing properties. Updates are needed under the following conditions:

   Changes in the enterprise architecture;

   Changes in known vulnerabilities;

   Changes in asset values;

   Changes in attacker profiles;

   Attacks or attack attempts occurring in the organisation;

   Attacks or attack attempts occurring in similar organisations;

   Changes in the organisational culture;

   Changes in the standard map components used in the model.

[4] A tool for data extraction from virtualised environments is explained in D2.2.2 [192]

[5] Note that this is not limited to only technical data. The Geographical Information System (GIS) in the ATM case study can provide automated extraction of heat maps in a certain area, which falls into the category of social data [193].

4. Alerts will be issued to the user when important updates have been made, or when user action is required to assist in the updates.
5. The user re-runs the analysis to identify any differences in the analysis results given the new conditions.
6. Alerts are issued when changes in the underlying data lead to significant changes in risk.
7. The model can be saved for future references or adjustments.

Required functionalities

- **Support for automatic updates and alerts or 'dashboard' (for Step 1)** [**U1.1**]
- **Support for automatic updates or 'news feed' (for Step 1)** [**U1.13**]
- Support for manual updates **[U1.14]**
- Support for alerts based on (manual or automatic) updates [**U1.15**]
- Model editing window: move, delete, and change properties of components [**U2.4**]
- (Re-)run analysis button [**U2.14**]

Use Case 4: Sensitivity Analysis

Functional goals/needs

The user wants to compute the effect of differences in the model on the risk values by making simulated modifications to the model or some of its parameters. The intended outcome is the difference in the outcome of the risk analysis that simulated changes would bring.

Description (workflow)

1. The user runs an analysis in TRESPASS.
2. **The user prepares for the sensitivity analysis. The user simulates modifications to the model where needed by changing properties of the desired model components.**
3. **(optional) The user saves different types of modifications (excluding the results) made for later comparison.**
4. The user re-runs the analysis in TRESPASS.
5. The system determines to what extent the result depends on changes in particular items.
6. The system keeps track of the different versions of the map, the changes (diff) between them, and the results of the analysis in terms of risk. The user can open a menu to select previously used maps/analyses, and select them to view the stored results.

Required functionalities

- Sensitivity analysis (changes in output based on changes in a particular parameter) [**U1.12**]
- **Summary of modifications simulated (for step 2 and 3)** [**TR5**]
- **Support for saving maps (for Step 6)** [**TR6**]
- **Support for automated generation of diff between different maps (for Step 6)** [**TR7**]

We have successfully identified seven additional requirements (**TR1** up to **TR7**) for the trace model in order to support seamless interaction between user and TRESPASS tools in conducting traceability-related activities. Similar to the requirements from the interview results, these Use Case-generated requirements can also be mapped against the goals. TR1 (*Support for additional submenu to insert plaintext input during model creation*) is closely related to G_{L2} (*The trace model should enable tracing of model components to the (recorded) decisions behind the modelling decisions (e.g. which components are included and which are abstracted out for what reasons; who made what decisions; what the context of the analysis is: when it was conducted, under what conditions and assumptions, etc.)*). TR2 (*Support for generating unique identifier for every model elements*) supports G_{L3} (*The trace model should enable users to pinpoint*

data throughout the TRE$_S$PASS process). TR3 (*Linking of model components as unique entities in different processes*) corroborates $G_{1.4}$ (*The trace model should enable tracing of analysis results back to the attack tree calculation, further back to the model components, and finally to see what data are involved*), and TR4 (*Support for flagging and manual annotation in the analysis results in plaintext format*) promotes the consummation of $G_{1.5}$ (*The trace model should support accountability in the risk assessment process and enable decision makers to defend security decisions made based on the risk assessment results*). Ultimately, the dynamic view goal ($G_{2.1}$) is advocated by U1.1 (*Support for automatic updates and alerts or 'dashboard'*), U1.13 (*Support for automatic updates or 'news feed'*), TR5 (*Summary of modifications simulated*), TR6 (*Support for saving maps*), and TR7 (*Support for automated generation of diff between different maps*).

6.1.3. List of requirements

Thus far we have generated a number of requirements from different processes, which may have resulted in overlap between one or more requirements. In this section, we produce a final list of requirements, classified into two different types: functional requirements (R_f) and user requirements (R_u). However, we also realise that there might be *non-functional requirements* that the trace model needs to possess. According to Wieringa, these requirements may include *utility, usability by a stakeholder,* and *interoperability with other systems* [221]. The final list of requirements is compiled by grouping several (overlapping) requirements under their common goal. This is shown in Table 6.2 as follows. Sources of these requirements are also given for clarity.

Table 6.2: Final list of requirements

Code	Requirement	Type of requirement	Associated goal	Source
TRM1	The trace model should be compatible with different types of data used in TRE$_S$PASS (e.g. physical data, social data, commercial data, etc.).	Non-functional requirement (compatibility)	$G_{1.1}$	Goals
TRM2	The trace model should be able to discriminate different types of data used in TRE$_S$PASS (e.g. vulnerability data, stakeholder goals, technical data, expert judgement, etc.).	Functional requirement	$G_{1.1}$	Goals
TRM3	The trace model should be able to discriminate multiple sources of data (e.g. data retrieved from data collection stage; data retrieved from databases such as vulnerabilities or scenarios, etc.).	Functional requirement	$G_{1.1}$	Goals
TRM4	The trace model should be integrated with the knowledge base as the "central data repository" and the TRE$_S$PASS model.	Functional requirement	$G_{1.1}$	Goals
TRM5	The trace model should be able to trace back data aggregation level used in the analysis.	Functional requirement	$G_{1.1}$	Interviews
TRM6	The trace model should be able to trace back how recent the datasets used in the analysis are.	Functional requirement	$G_{1.1}$	Interviews
TRM7	The trace model should be able to incorporate different types of modelling tools in the modelling process (e.g. ArgueSecure, LEGO model, manual drawing + meeting notes, etc).	Functional requirement	$G_{1.2}$	Goals
TRM8	The trace model should enable annotation in the manual modelling process leading up to the final model (e.g. place tags in drawings, insert actors in meeting notes, etc.).	User requirement	$G_{1.2}$	Goals
TRM9	The trace model should be integrated with the different libraries used in the model creation process (e.g. Model Pattern Library, Attacker Profile Library, etc.).	Functional requirement	$G_{1.2}$	Goals
TRM10	the trace model should support insertion of comments in plaintext format during model creation.	User requirement	$G_{1.2}$	Use Case 1
TRM11	The trace model should be able to uniquely identify data used in the risk assessment process by means of generating unique identifier.	Functional requirement	$G_{1.3}$	Goals & Use Case 1
TRM12	The trace model should be able to store intermediate parameters generated during the risk assessment process that are not currently stored in any of the TRE$_S$PASS tools.	Functional requirement	$G_{1.3}$	Goals & interviews

Table 6.2 Final list of requirements (continued)

Code	Requirement	Type of requirement	Associated goal	Source
TRM13	The trace model should be integrated with the whole TRE$_S$PASS process (from the data collection stage up to the visualisation stage).	Non-functional requirement (interoperability with other systems)	$G_{1.4}$	Goals
TRM14	The trace model should be aligned and enable support for different TRE$_S$PASS concepts (e.g. Actors, Policies, Locations, etc.).	Non-functional requirement (interoperability with other systems)	$G_{1.4}$	Interviews
TRM15	The trace model should enable linkage between the TRE$_S$PASS model and attack trees used in the analysis stage (both generic and annotated).	Functional requirement	$G_{1.4}$	Interviews
TRM16	The trace model should enable linkage between the TRE$_S$PASS model and tools that generate, visualise, and analyse the attack trees.	Functional requirement	$G_{1.4}$	Interviews
TRM17	The trace model should enable linkage between the TRE$_S$PASS model and its inputs (social data, physical data, and model templates).	Functional requirement	$G_{1.4}$	Interviews
TRM18	The trace model should enable linkage between the TRE$_S$PASS model and attacker profiles used.	Functional requirement	$G_{1.4}$	Interviews
TRM19	The trace model should enable linkage between the knowledge base and its input sources (Attacker Profile Library, Entity Type Library, online vulnerability databases, and binary technical data).	Functional requirement	$G_{1.4}$	Interviews
TRM20	The trace model should enable linkage between the knowledge base and the tools that augment, annotate, and analyse the attack trees.	Functional requirement	$G_{1.4}$	Interviews
TRM21	The trace model should enable linkage between the knowledge base and the annotated attack trees.	Functional requirement	$G_{1.4}$	Interviews
TRM22	The trace model should enable identification of attack patterns used in the annotated attack trees.	Functional requirement	$G_{1.4}$	Interviews
TRM23	The trace model should enable linkage between analysis results to the tools that generate and visualise the attack trees.	Functional requirement	$G_{1.4}$	Interviews
TRM24	The trace model should enable linkage between the visualised results back to the tool that visualise the attack trees.	Functional requirement	$G_{1.4}$	Interviews
TRM25	The trace model should be able to explicitly show how a unique entity is used in different TRE$_S$PASS processes.	Functional requirement	$G_{1.4}$	Use Case 1
TRM26	The trace model should enable explicit reasoning behind security decisions taken and risk assessment results by providing support for flagging and manual annotation in the analysis results in plaintext format.	User requirement	$G_{1.5}$	Goals & Use Case 2
TRM27	The trace model should be capable of storing the arguments behind security decisions to enable future reference.	Functional requirement	$G_{1.5}$	Goals
TRM28	The trace model should be able to automatically identify changes in the input data, alert the user, and trigger a re-analysis based on these changes.	Functional requirement	$G_{2.1}$	Goals
TRM29	The trace model should be able to identify changes made in input/model components in an analysis as compared to a prior analysis and produce a summary of simulated changes.	Functional requirement	$G_{2.1}$	Use Case 4
TRM30	The trace model should be able to store different analysis results to enable comparison and provide a log thereof.	Functional requirement	$G_{2.1}$	Goals, interviews, and Use Case 4
TRM31	The trace model should be able to show the differences between multiple analyses.	Functional requirement	$G_{2.1}$	Goals & Use Case 4

One may encounter some of the requirements devised above in a different expression in previous deliverables

in various work packages. Such requirements include R17, R43, R48, and R91 in D1.1.2 (Final specifications and requirements for socio-technical security models), R65 in D2.1.2 (Final requirements for the data management process), R68, R69, and R87 in D6.1.2 (Final requirements for tool integration), as well as R64, R113, R115, R116, and R119 in D6.2.2 (Final refinement of functional requirements). This overlap demonstrates that the idea to embed traceability has over time appeared in various work packages in TRESPASS but it needs more substantiation, which is precisely the goal of this thesis in general and this chapter in particular.

6.1.4. Assumptions and contribution arguments

The requirements listed above need to be backed up with necessary assumptions to contribute to the stakeholder goals. In this section we list the necessary assumptions affiliated with each requirement and express how the combination will contribute to the enactment of the stakeholder goals (in the form of contribution arguments). The assumptions vary according to the type of respective requirements they support. The contribution arguments are presented for each requirement in Table 6.3 according to the following formula:

(Artefact Requirements) x (Context Assumptions) contribute to (Stakeholder Goals)

Table 6.3: Contribution arguments

Requirement	Contribution argument
TRM1	**If** the trace model is compatible with different types of data used in TRESPASS **and** there is more than one type of data used in a typical risk assessment using TRESPASS, **then** the trace model enables tracing of data used in the analysis back to the respective data sources.
TRM2	**If** the trace model is able to discriminate different types of data used in TRESPASS **and** there is more than one type of data used in a typical risk assessment using TRESPASS, **then** the trace model enables tracing of data used in the analysis back to the respective data sources.
TRM3	**If** the trace model is able to discriminate multiple sources of data **and** there is more than one type of data source used in a typical risk assessment using TRESPASS, **then** the trace model enables tracing of data used in the analysis back to the respective data sources.
TRM4	**If** the trace model is integrated with the knowledge base and the TRESPASS model **and** the majority of data used for the risk assessment is stored in the knowledge base, **then** the trace model enables tracing of data used in the analysis back to the respective data sources.
TRM5	**If** the trace model is able to trace back data aggregation level used in the analysis **and** TRESPASS engages with different levels of data aggregation, **then** the trace model enables tracing of data used in the analysis back to the respective data sources.
TRM6	**If** the trace model is able to trace back how recent the datasets used in the analysis are **and** TRESPASS engages with types of data that can be superseded with more recent data ('live' databases), **then** the trace model enables tracing of data used in the analysis back to the respective data sources.
TRM7	**If** the trace model is able to incorporate different types of modelling tools in the modelling process **and** the modelling process in TRESPASS can be recorded and can be done with the help of modelling tools, **then** the trace model enables tracing of model components to the (recorded) decisions behind the modelling decisions.
TRM8	**If** the trace model enables annotation in the manual modelling process leading up to the final model **and** the models used in the modelling process can be 'digitised' with the help of appropriate tools, **then** the trace model enables tracing of model components to the (recorded) decisions behind the modelling decisions.
TRM9	**If** the trace model is integrated with the different libraries used in the model creation process **and** the modelling process in TRESPASS involves the use of different libraries available in TRESPASS toolkit, **then** the trace model enables tracing of model components to the (recorded) decisions behind the modelling decisions.
TRM10	**If** The trace model supports insertion of comments in plaintext format during model creation **and** additional information that cannot be captured with existing tools is needed during model creation to give additional explanation on the modelling decisions and their context, **then** the trace model enables tracing of model components to the (recorded) decisions behind the modelling decisions.
TRM11	**If** the trace model is able to uniquely identify data used in the risk assessment process by means of generating unique identifier **and** each data used in a risk assessment activity in TRESPASS is unique (no duplicate), **then** the trace model enables users to pinpoint data throughout the TRESPASS process.
TRM12	**If** the trace model is able to store intermediate parameters generated during the risk assessment process that are not currently stored in any of the TRESPASS tools **and** there are intermediate parameters generated during the risk assessment process (e.g. probability of attack, risk, attack time), **then** the trace model enables users to pinpoint data throughout the TRESPASS process.

Table 6.3 Contribution arguments (continued)

Requirement	Contribution argument
TRM13	**If** the trace model is integrated with the whole TRE$_S$PASS process (from the data collection stage up to the visualisation stage), **then** the trace model enables tracing of analysis results back to the attack tree calculation, further back to the model components, and finally to see what data are involved.
TRM14	**If** the trace model is aligned with different TRE$_S$PASS concepts (e.g. Actors, Policies, Locations, etc.) and is able to support them, **then** the trace model enables tracing of analysis results back to the attack tree calculation, further back to the model components, and finally to see what data are involved.
TRM15	**If** the trace model enables linkage between the TRE$_S$PASS model and attack trees used in the analysis stage (both generic and annotated) **and** in a typical risk assessment using TRE$_S$PASS the attack trees use information from the TRE$_S$PASS model, **then** the trace model enables tracing of analysis results back to the attack tree calculation, further back to the model components, and finally to see what data are involved.
TRM16	**If** the trace model enables linkage between the TRE$_S$PASS model and tools that generate, visualise, and analyse the attack trees **and** in a typical risk assessment using TRE$_S$PASS the attack trees make use of information stored in the knowledge base, **then** the trace model enables tracing of analysis results back to the attack tree calculation, further back to the model components, and finally to see what data are involved.
TRM17	**If** the trace model enables linkage between the TRE$_S$PASS model and its inputs (social data, physical data, and model templates) **and** in a typical risk assessment using TRE$_S$PASS the tools that involve attack trees calculation make use of information stored in the knowledge base, **then** the trace model enables tracing of analysis results back to the attack tree calculation, further back to the model components, and finally to see what data are involved.
TRM18	**If** the trace model enables linkage between the TRE$_S$PASS model and attacker profiles used **and** in a typical risk assessment using TRE$_S$PASS, the TRE$_S$PASS model uses information from Attacker Profile Library, **then** the trace model enables tracing of analysis results back to the attack tree calculation, further back to the model components, and finally to see what data are involved.
TRM19	**If** the trace model enables linkage between the knowledge base and its input sources (Attacker Profile Library, Entity Type Library, online vulnerability databases, and binary technical data) **and** in a typical risk assessment using TRE$_S$PASS the attack trees make use of information stored in the knowledge base, **then** the trace model enables tracing of analysis results back to the attack tree calculation, further back to the model components, and finally to see what data are involved.
TRM20	**If** the trace model enables linkage between the knowledge base and the tools that augment, annotate, and analyse the attack trees **and** in a typical risk assessment using TRE$_S$PASS the tools that involve attack trees calculation make use of information stored in the knowledge base, **then** the trace model enables tracing of analysis results back to the attack tree calculation, further back to the model components, and finally to see what data are involved.
TRM21	**If** the trace model enables linkage between the knowledge base and the annotated attack trees **and** in a typical risk assessment using TRE$_S$PASS the annotated attack trees make use of information stored in the knowledge base, **then** the trace model enables tracing of analysis results back to the attack tree calculation, further back to the model components, and finally to see what data are involved.
TRM22	**If** the trace model enables identification of attack patterns used in the annotated attack trees **and** in a typical risk assessment using TRE$_S$PASS the annotated attack trees make use of attack patterns (either stored in Attack Pattern Library or custom-made), **then** the trace model enables tracing of analysis results back to the attack tree calculation, further back to the model components, and finally to see what data are involved.
TRM23	**If** the trace model enables linkage between analysis results to the tools that generate and visualise the attack trees **and** in a typical risk assessment using TRE$_S$PASS the analysis results logically follow from the attack trees generated with Tree Maker and visualised by ADTool, **then** the trace model enables tracing of analysis results back to the attack tree calculation, further back to the model components, and finally to see what data are involved.
TRM24	**If** the trace model enables linkage between the visualised results back to the tool that visualise the attack trees **and** in a typical risk assessment using TRE$_S$PASS the visualised results logically follow from the attack trees visualised by ADTool, **then** the trace model enables tracing of analysis results back to the attack tree calculation, further back to the model components, and finally to see what data are involved.
TRM25	**If** the trace model is able to explicitly show how a unique entity is used in different TRE$_S$PASS processes **and** in a typical risk assessment using TRE$_S$PASS a unique data entity is visible/identifiable from the data collection stage up to the analysis stage, the visualised results logically follow from the attack trees visualised by ADTool, **then** the trace model enables tracing of analysis results back to the attack tree calculation, further back to the model components, and finally to see what data are involved.
TRM26	**If** the trace model enables explicit reasoning behind security decisions taken and risk assessment results by providing support for flagging and manual annotation in the analysis results in plaintext format **and** the security decisions logically follow from the risk assessment results, **then** the trace model supports accountability in the risk assessment process and enable decision makers to defend security decisions made based on the risk assessment results.

Table 6.3 Contribution arguments (continued)

Requirement	Contribution argument
TRM27	**If** the trace model is capable of storing the arguments behind security decisions to enable future reference **and** the security decisions may need to be defended in the future in case things go in an unexpected direction, **then** the trace model supports accountability in the risk assessment process and enable decision makers to defend security decisions made based on the risk assessment results.
TRM28	**If** the trace model is able to automatically identify changes in the input data, alert the user, and trigger a re-analysis based on these changes **and** a pre-specified condition of changes to trigger re-analysis is configured, **then** the trace model is able to support propagation of changes in the input data/model components throughout the analysis process.
TRM29	**If** the trace model is able to identify changes made in input/model components in an analysis as compared to a prior analysis and produce a summary of simulated changes **and** the changes made are valid, **then** the trace model is able to support propagation of changes in the input data/model components throughout the analysis process.
TRM30	**If** the trace model is able to store different analysis results to enable comparison and provide a log thereof **and** one analysis result does not render the previous ones invalid and analysis results do not overwrite one another, **then** the trace model is able to support propagation of changes in the input data/model components throughout the analysis process.
TRM31	**If** the trace model should be able to show the differences between multiple analyses **and** the changes made are valid, **then** the trace model is able to support propagation of changes in the input data/model components throughout the analysis process.

6.1.5. Design criteria

Design criteria are results of unravelling and operationalisation of requirements into more measurable terms [213]. As we only have two types of requirements, we also expect only two types of design criteria although the distinction of the two may not be crystal clear. Operationalisation in this sense is defined as specifying involved tools, concepts, and processes that are attached to each requirement. From this operationalisation, more precise and design-oriented imperatives are formulated, which we name design criteria. They can be found in Table 6.4 below.

Table 6.4: Design criteria

Requirement	Design criterion/criteria	Criterion code
TRM1	The trace model should implement TRE$_S$PASS data management process, which includes data transformation and data manipulation.	CR1
TRM2	The trace model should support metadata of data type, its value, and its domain (e.g. technical or social).	CR2
TRM3	The trace model must contain metadata elements used to classify data to allow identification of the data source, creator of the data and who modified the data	CR3
TRM4	The trace model should support (1) consolidation process of different data types onto the knowledge base (e.g. data extracted from virtualised infrastructures [192] and socio-technical data) and (2) transformation of data into the TRE$_S$PASS base model[6]	CR4
TRM5	The trace model should support data classification by providing metadata of data levels of measurement (e.g. nominal, ordinal, interval, or ratio).	CR5
TRM6	The trace model should support control metadata of creation and last updated timestamps and status (Active, Inactive, Removed).	CR6
TRM7	The trace model should support transformation/embedding of the output of non-formalised modelling tools, such as ArgueSecure arguments, pictures of LEGO models, and pictures of tangible models. This can be achieved for instance by enabling a feature to upload pictures or text files.	CR7
TRM8	The trace model should be able to especially identify elements of non-formalised model development tools such as ArgueSecure (Assets), LEGO (rich picture of Actors, infrastructure, data, data flow, and Locations) and the tangible modelling toolkit (Actor, Assets, Location, Access policy, and Relationship) through tagging.	CR8
TRM9	The trace model should be able to identify model elements derived from libraries (template and model elements from Model Pattern Library, attacker profiles from Attacker Profile Library, and annotated attack trees from Attack Pattern Library).	CR9

[6]Part of this is currently already supported. When a user creation creates a model in ANM, the tool interacts with knowledge base to establish a model context with a *modelID* [199].

Table 6.4 Design criteria (continued)

Requirement	Design criterion/criteria	Criterion code
TRM10	The trace model should be integrated with the ANM as the central tool in the model creation stage and should support (1) insertion of generic comments for the model as a whole and (2) annotation of the model's nodes and edges for additional explanation.	CR10
TRM11	The trace model should allow unique identification for each unique piece of data.	CR11
TRM12	The trace model should have a data staging area that stores parameters into a local persistence data model.	CR12
TRM13	The trace model should implement TRE$_S$PASS data management process, which includes data transformation (mapping and transformation modules) and data manipulation (input, manipulation, and output modules).	CR13
TRM14	The trace model should be aligned with the TRE$_S$PASS base model which includes the base concepts used in TRE$_S$PASS [7]	CR14
TRM15	The trace model should support XML format as the main format used in TRE$_S$PASS model and attack trees. It should be able to identify Locations, Edges, Assets, Actors, Predicates, Policies, and Processes in the model. [8]	CR15
TRM16	The trace model should be able to link Locations, Edges, Assets, Actors, Predicates, Policies, and Processes in the attack trees generation and analysis back to the model. The trace model should be able to connect visualised attack paths to the parameters generated during the analysis (difficulty, cost, time, probability).	CR16
TRM17	The trace model should be linked with the ANM as the main tool in creating the TRE$_S$PASS model and help generate IDs for data imported into the tool or library elements imported into the model.	CR17
TRM18	The trace model should generate unique Actor IDs for attacker profiles imported into the ANM.	CR18
TRM19	The trace model should be able to provide logger for the knowledge base that stores original data received and its associated metadata (timestamps, origins, version, etc).	CR19
TRM20	The trace model should support leaf expansion in APLTool by linking Actors and Assets in the attack tree with knowledge base data. The trace model should also further be able to link these Actors and Assets with quantitative attack properties such as attack difficulty, skill level, or likelihood.	CR20
TRM21	See criterion of TRM20.	CR20
TRM22	The trace model should be able to generate unique identification for attack patterns used in the annotated attack trees.	CR22
TRM23	The trace model should enable linkage of quantitative properties resulting from the analysis of attack trees (e.g. attack difficulties, cost of attack, etc.) to the original attack trees generated with Tree Maker and visualised in ADTool.	CR23
TRM24	The trace model should be able to link nodes in the navigator map to the leaves of the attack tree as visualised in the ADTool.	CR24
TRM25	The trace model should enable callback of any piece of data by recalling its identifier and show where it shows up in the risk assessment process.	CR25
TRM26	The trace model should support plaintext format input in the navigator map in the ANM.	CR26
TRM27	The trace model should enable storage of the plaintext input in the navigator map in the ANM (in .txt format).	CR27
TRM28	The trace model should be able to notify users based on push notifications of updates on external data sources or automated data extraction tools (facilitated through the knowledge base) and prompt the user whether to conduct a re-analysis or not.	CR28
TRM29	The trace model should be able to show differences across different model components in terms of TRE$_S$PASS concepts (e.g. Locations, Assets, Policies, etc.) and show the identifier of the different elements to enable easy comparison.	CR29
TRM30	The trace model should enable storage of analysis results and generate metadata that enables unique identification of each analysis result (e.g. timestamp, owner, version, etc.)	CR30

[7] According to D2.1.2, TRE$_S$PASS base model consists of base concepts, attack definitions, and attacker goals and profiles [191].

[8] According to D3.4.1, an attack is an act of policy invalidation which involves a Policy to be invalidated, the Actor who performs the attack, the actions to perform, the Location in which the attack is carried out, and the Assets to be compromised in the attack. [185]. The same deliverable also features descriptions of these components in XML format.

Table 6.4 Design criteria (continued)

Requirement	Design criterion/criteria	Criterion code
TRM31	The trace model should be able to show the differences across analyses in terms of quantitative properties (e.g. difficulty, cost, time, likelihood).	CR31

The current integration effort in TRE$_S$PASS includes placing the knowledge base as the 'data center' in TRE$_S$PASS, meaning that a large part of the information system in TRE$_S$PASS will reside in the knowledge base [199]. Some of the design criteria for the trace model can already be supported by this campaign.

6.1.6. Evaluation of requirements, assumptions, and design criteria

At this stage, we evaluate whether the outputs of the current process are still aligned. We follow the evaluation procedures (plan, process, and product) as put forth by Verschuren & Hartog [213] by running checks on the following items:

- Empirical validity, reliability, researcher independence and verifiability of the results that lead to the formulation of R_u;
- Whether R_f covers G without any errors of omission or errors or commission;
- The logic of the combination of and the relations between R_f *and* R_u;
- The methodological guidelines in deriving R_f, and R_u and translating them to the various design criteria;
- Whether the assumptions are credible enough; and
- Whether the design criteria have been operationally defined, and whether they cover the requirements and match the goals.

As mentioned early in this chapter, the formulation of requirements essentially involved three primary sources: the goals as have been formulated in Chapter 5, parts of the interview results, and Use Cases of traceability-supported TRE$_S$PASS . We trust that this was a manifestation of the *triangulation* principle which in turn ensured the empirical validity and the reliability of the requirements. The empirical dimension was embodied in the interviews, in which opinions of experts and researchers who have had hands-on experience with developing various parts of TRE$_S$PASS were collected. This guaranteed a high level of relevance with our research. Researcher independence and verifiability in deriving the requirements were iteratively examined by the research supervisor.

Another important aspect to evaluate is whether there is coherence between the functional requirements and the goals. This is important since goals and functional requirements should form a goal-subgoal relationship to guarantee a correct cascading process until the design idea is substantiated in the form of a final product. Common errors at this phase include errors of commission (adding more than necessary) and errors of omission (leaving out the necessary) [213]. We have tried to avoid the occurrence of these errors by systematically connecting each requirement to its associated goal. This was also done to help the reader understand the links between the requirements and the goals. For the same reason, we grouped relevant functional requirements and user requirements under their common goal.

We also examine the contribution arguments, namely the logic behind every requirement and its underlying assumption towards the fulfilment of the goal. At least one assumption exists for every functional requirement and user requirements. Non-functional requirements are not accompanied with assumptions. The credibility of these assumptions are once again scrutinised by the research supervisor. Lastly, operationalisation of design criteria are done by specifying the tools and processes that would be involved in the effort of fulfilling a certain requirement. This has been done by referring to both internal and public deliverables that describe the various processes TRE$_S$PASS. We deliberately did not make the choice of expressing the design criteria in modalities or scores, because that would make the design even more abstract. For instance, expressing the requirement to 'enable linkage between X and Y' in numerical values would result in a vague design criteria. It would be more useful to decompose it into the tools that are involved and the processes that contribute to the fulfilment of the requirement.

6.2. Identifying available treatments

As TRESPASS is a state-of-the-art approach towards assessing socio-technical risks, we do not expect any readily available treatments from other approaches that can be applied to TRESPASS to handle the need for traceability. However, in this section we evaluate current approaches to handle traceability from various risk assessment methods. To facilitate an apples-to-apples comparison, the reach of evaluation here is exclusively kept to tool-based approaches. Readers should not expect to find discussion on OCTAVE or FAIR, as both of them are not tool-assisted and are fairly driven by manual efforts.

Perhaps the closest attempt to maintain traceability in a risk assessment process is manifested in CORAS trace model. However, to begin with, CORAS does not have an explicit formulation of the idea of the traceability. Instead, CORAS places heavy emphasis on the idea of *change* [117] and *evolution* [116]. Following our definition in Chapter 4, traceability is implicitly interpreted solely in its dynamic sense. To the best of our knowledge, the idea of traceability in CORAS does not account for the static view. CORAS however extensively discusses dependencies [115], which amounts to the documentation and reasoning of assumptions in the CORAS process. According to Solhaug and Seehusen, establishing traceability with regards to changes in CORAS assisted by the *trace model* is done by conducting the following activities [171]:

1. First iteration (initialisation): Conducting the initial risk analysis, constructing the target model (or the system model) and the risk model, and compiling the trace model that captivates the relationship between the target model and the risk model. These processes are captured in two core CORAS activities: *establishing the context* and *identifying risks*; and

2. Continuous risk analysis: Capturing the changes in the context of the analysis and the target of analysis, and identifying and documenting changes to the risk model as aftermath of the changes in the target of analysis. These processes involve slightly modified CORAS activities: *establishing the* ***changed*** *context* and *identifying the* ***changed*** *risks*.

The paper further explicates the highlights of the trace model-assisted approach to support traceability in CORAS as follows [171]:

- It is based on international standards, namely ISO 31000 standard on risk management;
- It offers a complete package, ranging from processes and guidelines to the necessary tool support;
- It allows for on-the-fly modelling of changes;
- It facilitates different languages for target modelling;
- It grants traceability between target model and risk model;
- It helps users to identify changes in the target of analysis speedily;
- It supports change-impacted risks in the target system;
- It can identify and solve inconsistencies in the analysis; and
- It enables rapid risk re-analysis.

The primary artefacts involved in this approach can be found in Figure 6.2. The rest of the CORAS process is indifferent with regards to the continuous risk analysis. Only the first two steps reflect the involvement of the trace model in giving traceability a foothold in CORAS. The trace model itself serves to relate the elements of a *generic target model* in CORAS to elements of the risk model. A trace model is formally defined as a set of pairs containing an element of a risk model r and an element of the target model t (r,t).

Seehusen and Solhaug also mention other instances of tool-supported efforts to model dependencies in risk analysis [164], for example ProSecO, dependent CORAS, Rinforzando, and SOA Modelling Suite (SMS). ProSecO emphasises heavily on enterprise architectures as the basis of the risk analysis and peruses the dependencies of the system, but not dependencies between the processes involved [88]. Our notion of traceability seeks to explore the latter (cf. the definition given at the end of Chapter 4). Moreover, the approach is more qualitative and consequently does not involve calculations of risk. Risk is rather only depicted in a bird's eye view as the combination of either low, medium, or high likelihood times impact. Dependent CORAS is more focused towards interdependencies between

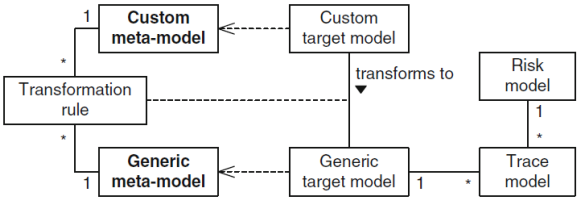


Figure 6.2: Primary artefacts in the CORAS traceability approach

Source: [171]

different risk models. Our proposed trace model for TRESPASS contributes by providing an ambidextrous[9] view on traceability, encompassing both a static view (commonly known as dependencies) and a dynamic view (commonly known as evolution or changes) of traceability in a risk assessment process.

6.3. Carrying out the design process

According to the interview results, an ideal representation of the trace model would take the form of a tool support. However, the available time does not allow us to develop a fully mature tool support that is ready to be embedded into TRESPASS toolchain. A more feasible solution is to specify the processes that the trace model would be able to execute to fulfil its function and demonstrate how it would fit in within the already existing TRESPASS tools. The goal of this section is to transparently show how design criteria (*C*) can be translated into structural specifications (*S*).

Because developing a tool support is out of the question for this thesis, we look at other options to extend TRESPASS such that traceability can be supported. More possible forms of extension include language extension, tool extension, and internal embeddings. These choices were consciously made by levering the existing features of the integrated TRESPASS process, including:

1. TRESPASS has a data management process and requirements that can be used as touchstones [191]. These processes/ requirements are, for instance:

   Addition of metadata elements to support data elements during storage operation, especially control metadata;

   Extracted data should be worked into a database model which has a standard section (fixed) identification, for example particular fields; and

   Data consolidation requires the identification of the data source domain and the origin of each piece of data.

   The limiting part of these processes/requirements is that not all (control) metadata can be automatically manufactured, especially when it comes to social data. A quick example is when the user relies on expert judgement for a specific piece of data. The user will need to clearly identify the source of this data (Who is the expert? Where is the data recorded? etc.) User effort is mandatory to have a comprehensive control metadata.

2. The Attack Navigator Map (ANM) is positioned as the main user interface and central tool in the TRESPASS toolchain. All the tools are integrated and accessible through the ANM [196]. TRESPASS model components also work in terms of the ANM interface. Traceability can be supported by adding user-facing features into the ANM (tool extension).

3. There are a few options to extend TRESPASS: tool support, language extension, and internal embeddings. Some of the proposed approaches suggest extension through internal embeddings and language extension, especially *low-level embeddings* (translating model extension to elements that exist in the original modelling language) [189].

   Another possible alternative is to have *high-level embeddings* in which the language is enlarged with new constructs. However, there is a strong trade-off that must be made. With low-level embeddings, the model does not preserve any information. In turn, analyses and visualisations will not be able to work with them.

[9] An ambidextrous person is able to utilise both his/her left and right hand equally well.

As with high-level embeddings, many changes have to be made including specification for semantic rules and enhancement of model representations with new elements [189]. An example of this is to add new actions for the Actors or add new properties to the objects. In our specifications, we utilise *low-level embeddings* with the consideration that our specifications do not require further analyses and visualisations. What we propose is to simply connect existing elements in the current modelling language to form a tighter bond between processes.

4. The attack tree in TRESPASS has its own schema definition (`adtree.xsd`from ADTool), which allows explicit reasoning [184]. Every node in the attack tree contains a string sub-element `label` which acts as a label for the node. A number of specifications we outline involve modification of the label. Modification of `label` as a string variable that allows parsing for identification of model elements is investigated further in Chapter 7.

5. Knowledge base is positioned as the central repository within the TRESPASS process that interacts with most of the tools. This offers an advantage for storage of new data that come along with the specifications. For consistency, we advise that every generated traceability-related information (metadata, label, timestamp etc.) is stored altogether with the related instance (e.g. model persistence, TRESPASS Model File, Attack Tree, etc.). More information will be available in D2.4.1 which extensively discusses the TRESPASS Information System.

6. TRESPASS uses XML as a common language for the tools to communicate with. This minimises the risk of language incompatibility after addition of extensions.

We now run the attack scenarios depicted in Chapter 5 to exposit the processes that have to be executed by the trace model in TRESPASS in order to establish traceability in the TRESPASS process. To do this, selected Use Cases will be demonstrated to display the tasks that will have to be performed by the user. Whenever possible, we show actual screen captures of the TRESPASS tools. Following the illustration of the processes, we can derive the structural specifications of the trace model. Where applicable, we link the processes in the Use Cases to the navigation metaphor or the service model as explained in D5.4.2 [199] to provide better context. There are four steps involved in the service model as follows:

1. **Satellite view**

   Creating an informal/semi-formal pictorial visualisation of the system of interest, accompanied with possible attacker profiles. This is done through among others, LEGO modelling activity or other tangible modelling interaction. In the integration diagram, this would correspond to the data collection stage.

2. **Map**

   Transforming the satellite view representation into a map containing elements of the TRESPASS concepts, such as Actors, Policies, and Locations. The map should also already able to depict the attackers' formal properties, such as the budget they have in hand to perform attacks. The integration diagram captures this stage in the model creation stage.

3. **Routes**

   Calculating the possible routes that the attacker might take in order to arrive at their 'destination' by making use of TRESPASS analysis tools. The analysis stage and visualisation stage in the integration diagram are covered in this stage.

4. **Optimisation**

   Incorporating defensive measures into the initial map and re-running the analysis to examine the effectiveness of countermeasures in the system. This would be an example of a dynamic risk assessment as laid out in Chapter 4.

Readers should bear in mind that not all steps in the navigation metaphor can be explained in the Use Cases, as every Use Case is intended to draw focus towards different support capabilities of traceability. To help readers better understand Scenario 1, we choose to combine Use Case 1 (Understanding Analysis Results) and Use Case 2 (Security Investment) in order to demonstrate the static view functionalities of traceability. This choice was made because the first five steps of both Use Cases are identical and therefore we only need to incorporate the latter halves. However it is very likely that the Optimisation stage is excluded in this demonstration, as this concerns reanalysis which would fit better in a dynamic view Use Case. We first explain Scenario 1 from Chapter 5 in the combined Use Case while also drawing references from the navigation metaphor, and subsequently Scenario 2 in Use Case 3.

6.3.1. Scenario 1 applied to Use Case 1 and 2

In this section, the steps describing how Use Case 1 can be carried out will be explained in detail. We also draw references to actual TRESPASS tools and concepts.

The user opens the interface, which shows an empty map of the organisation. The user inserts several standard elements onto the map and draws the relevant connections. If an Archimate model already exists, this may be used as a basis instead.

Before starting, we assume that there already exists a model of the organisation. In other words, the *satellite view* representation has been already constructed through one of the many available modelling tools and processes offered in TRESPASS [194]. This also means that the TRESPASS model is not spontaneously created from scratch in the ANM. The integration of the modelling processes with the model creation stage in TRESPASS should be done by considering several design criteria we have previously distinguished, namely CR7 and CR8. We proceed by giving their specifications as follows.

- **CR7: The trace model should support upload of the output of the non-formalised modelling tools, such as ArgueSecure arguments[10], pictures of LEGO models, and pictures of tangible models).**

  Specification: During the model creation in ANM, there should exist an additional option that enables upload of attachments to allow users to enclose the results of the manual modelling process. This option should be embedded within the sub-wizard[11] **Import**, which currently only has two options: **Import model** (to import an XML format model) and **Import floorplan** (to import the floorplan of the target of analysis). A third option called **Import manual model** could be appended to this sub-wizard. The attachment should be stored together with the TRESPASS model with which it is linked.

  Example: Users can upload the spreadsheet containing the ArgueSecure arguments, for instance as shown in Figure G.1 or a photo of the tangible modelling activity.

- **CR8: The trace model should be able to especially identify elements of non-formalised model development tools such as ArgueSecure (Assets), LEGO (rich picture of Actors, infrastructure, data, data flow, and Locations) and the tangible modelling toolkit (Actor, Assets, Location, Access policy, and Relationship) through tagging.**

  Specification: In the case of a picture attachment, 'tagging' is made possible by (right-)clicking on an element of the picture and associating it with a specific TRESPASS model element (very similar to photo tagging feature in social media platforms such as Facebook or Instagram). Upon import of the model through the **Import manual model** option, user will be prompted to tag the picture as necessary. This can be done *ex-ante* after all model elements in the TRESPASS model have been spawned. For ArgueSecure arguments (see Appendix G.2, annotations could be made by inserting the ID and Name of Assets involved in the rightmost column of the ArgueSecure template (again, after the TRESPASS model components have been generated so that their `id` and `label` become known). In this case, modifications to the attachment should be a default feature in the ANM (within the **Import manual model** sub-wizard). The attachments should also be stored in the knowledge base, preferably with an automatically-generated file name that allows easy corresponding to the TRESPASS model. This way, tracebility between the TRESPASS model and the results of the manual modelling process can be established.

  Example: An example of a tagged picture of the tangible model is given in Figure 6.3.

Results of such modelling activities may grow into a high-level model of the organisation as provided in Figure 5.3. After the high-level overview has been agreed upon, the model is transformed into a map of the organisation in the ANM, wherein every elements involved and every relevant connections are included. This demarcates the transition to creating the *map* from the satellite view. As of now, there is no official document yet that visualises the map view of the cloud case study. For solely illustration purposes, a *preliminary* map view of the model components is shown in Figure 6.4. Readers should keep in mind that this is not the final version. The key takeaway from this figure is that within the map, every possible connection should be included to be fed forward into calculation of the *routes*.

[10] Interested readers may wish to get their hands on the first version of ArgueSecure at `http://arguesecure.ewi.utwente.nl/login`. A stepwise guide to use it is available at `https://github.com/hitandyrun/arguesecure-online`. Ionita also discusses this in a scientific paper [91].

[11] To maintain consistency with the terminologies used for the ANM, we use the term 'wizard' to describe an interactive walkthrough of the tool during which the user receives guidance on how to complete the map [196]. The wizard is composed of different sub-wizards with specific functionalities. The list of sub-wizards is displayed in Appendix F

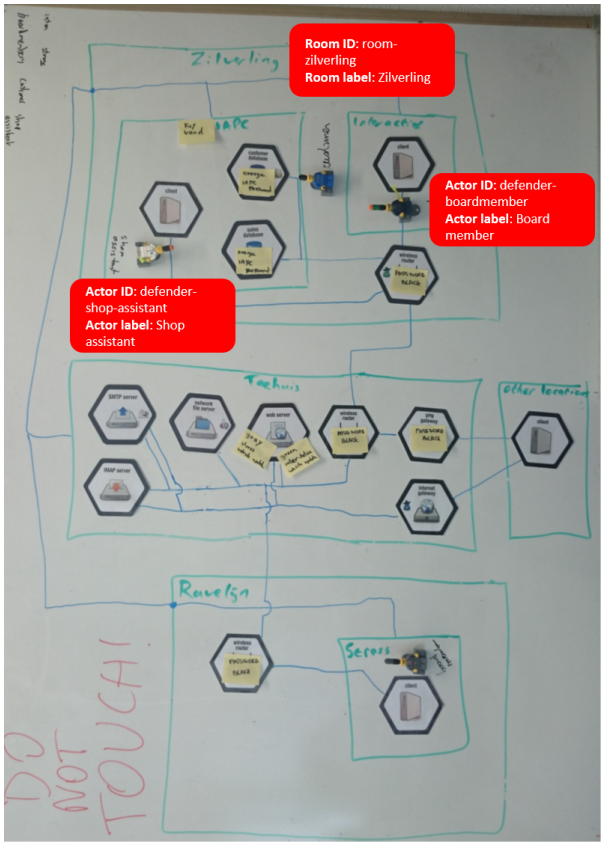


Figure 6.3: Example of a tagged attachment

Source: modified from [92]

Figure 6.4 shows that there exist various types of connections, for example access (`has_physical_access_to` and `has_logical_access_to`), working relationships and hierarchy (`contracted_by` and `manager_of`), or containment (`is_at` and `contained_in`). This naturally depends a lot on the types of elements that are included in the model. Note that the connections depicted here are limited to normal conditions. For example, of course Mr. Big can also have logical access to the Laptop if he asks Sydney to provide one, but it is excluded here as Mr. Big does not work with the Laptop on a daily basis. These kinds of assumptions are important to record, and the third step shows how such decisions can be recorded for traceability purposes. The attachments uploaded should be stored in the knowledge base along with the TRESPASS Model File instance.

The user clicks on the items to select the corresponding properties, which may consist of asset values, potential damage upon failure, access control mechanisms, and an estimation of their strength. Default values can be used if information is unavailable.

After the map is constructed in the ANM, the user should decorate the entities with relevant parameters. An initial suggestion of all the parameters that are attached to each entity is given in Figure 6.5. During the course of the project development several decisions to simplify these parameters have been made. However, there exists no official documentation for this yet. Similar to Figure 6.4, this is shown solely for illustration purposes only and not to be literally taken as the final version as the entities' parameters in TRESPASS. Some parameters might not end up in the final version of the entity description and there could be new parameters added in the final version.

Almost all of the components are equipped with the capability to store a unique *id* and a unique *label*. This is of paramount importance in facilitating traceability. The unique *id* and *label* allow the system to keep track of each entity as they travel along the TRESPASS process. We give examples for some of the components as follows. Other examples can be found in Appendix G.

- `actor`

  `id`: defender-big

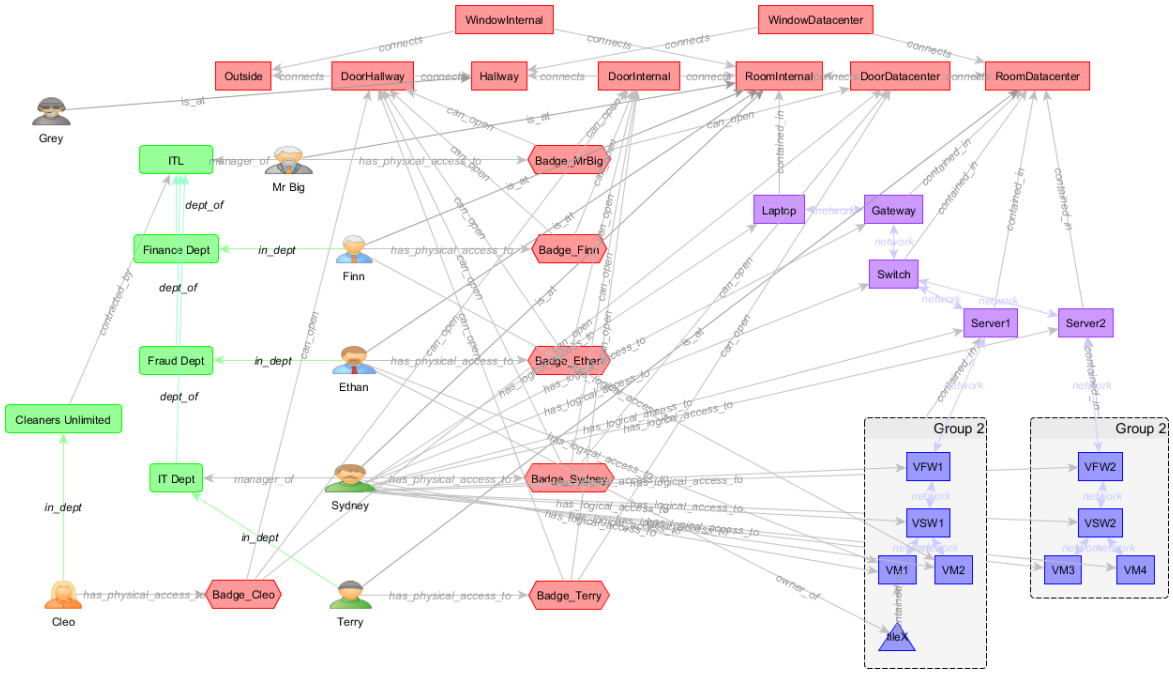


Figure 6.4: Map of ITL with relevant connections

Source: The TREsPASS project repository (modified)

label: Big

susceptibility_se[12]: High[13]

value_system[14]: N/A

age: 57

gender: Male

nationality: British

damage_reputation: Very high

personal_situation[15]: Unknown

- file

  id: fileX

  label: File X

  value: $ 500,000[16]

  size: 250 MB

  damage_financial: $ 500,000

  damage_reputation: Very high

- laptop

[12] This parameter denotes the susceptibility of the actor to a social engineering attack

[13] See assumptions of Scenario 1

[14] This parameter captures the possible values a person could have: for example being rule-abiding or would compromise corporate rules for friendship, or being open to the highest 'bidder'. Since it would be very complex to represent these kinds of values that a person could have, the current development excludes this parameter.

[15] This parameter captures a situation that a person experiences right now, for example in need of money or having a relative in jail. Due to its complexity, this parameter has been dropped from the current model.

[16] This is an estimation.

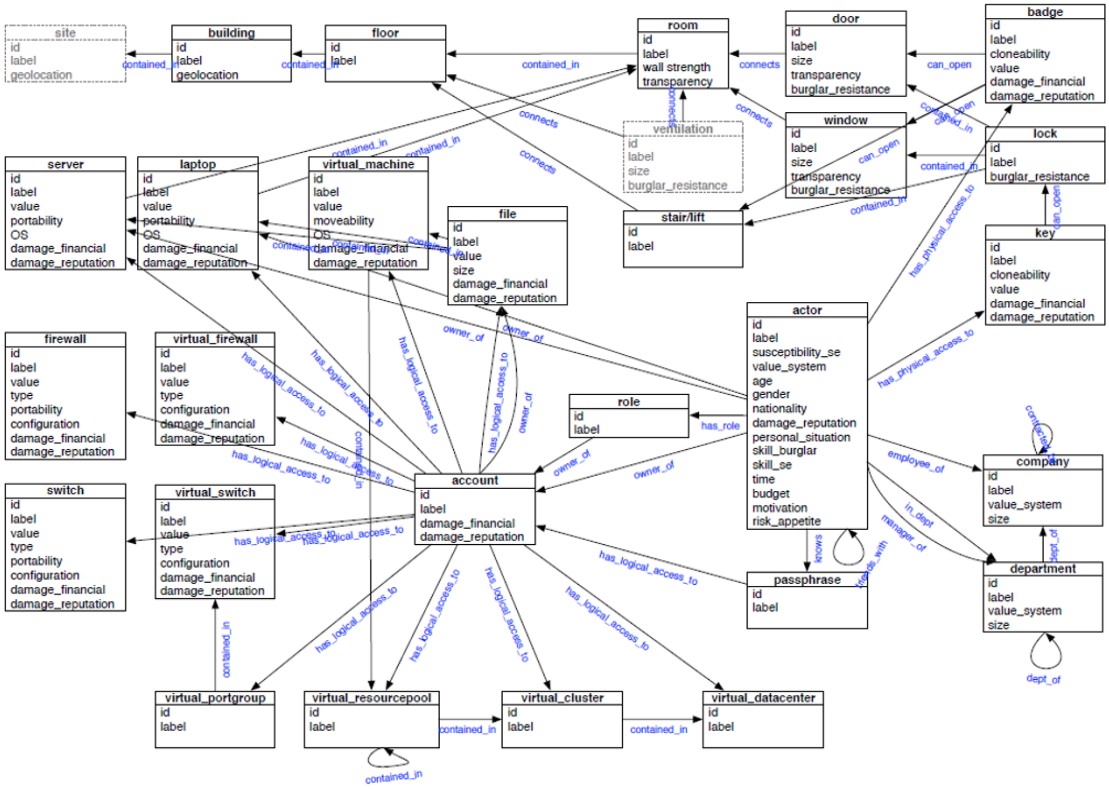


Figure 6.5: Attributes and parameters of the TRESPASS model components

Source: The TRESPASS project repository

`id`: laptop-admin

`label`: laptop

`value`: $ 500,000[17]

`portability`: Limited[18]

`OS`: Windows 10

`damage_financial`: $ 500,000

`damage_reputation`: Very high

These parameters can be treated as data that can be used to support traceability. However, *control metadata* can be added to install even greater level of traceability [191]. These metadata would reside within the data record level, enabling the user to trace back how the value for each parameter has been derived. Revisiting the criteria on data management (CR2, CR3, CR4, CR5, CR6, and CR19) from Table 6.4 brings us to the following yardstick for structural specifications involving metadata:

- **CR2: The trace model should support metadata of data type, its value, and its domain (e.g. technical or social).**

  Specification: Additional metadata field to identify the type (`data_type`) and domain (`data_domain`) of data can be added to relevant elements. As there are limited types of data, input to the `data_domain` field could be provided in a dropdown menu style. However, it is advised to provide a free text field to indicate the `data_type` field, as there could be endless possibilities to what the data type would be in different elements.

[17] At the very minimum, as ***fileX*** resides in this laptop.

[18] This is a company laptop assigned to Sydney, and only Sydney has the rightful privilege of carrying it around.

Example: As a running example, we will zoom into the `susceptibility_se` data embedded in component `actor`. We incrementally add the metadata fields as we progress through the ensuant criteria.

`susceptibility_se`: High

`data_domain`: **Social data**

`data_type`: **Experiment data**

- **CR3: The trace model must contain metadata elements used to classify data to allow identification of the source data domain, creation and last updated owner, and from where each single piece of data came.**

  Specification: Additional metadata fields to identify the source of data (`data_source`) that allows free text input; owner of the data as well as its modifier (`created_by` and `last_updated_by`). The `created_by` field should be non-modifiable, and the `last_updated_by` field should not overwrite the previous entry to allow full tracking of edit history. If the TRESPASS toolchain allows association with a unique user (through a unique username or login ID), the `created_by` and `last_updated_by` fields may be filled in automatically. These metadata fields should be added to every data field that requires user input.

  Example:

`susceptibility_se`: High

`data_domain`: Social data

`data_type`: Experiment data

`data_source`: **Brighton experiment 16/09/2015**

`created_by`: **ethanitl**[19]

`last_updated_by`: **ethanitl**

- **CR4: The trace model should support (1) consolidation process of different data types onto the knowledge base and (2) transformation of data into the TRESPASS base model**

  Specification: Metadata of TRESPASS model components should be stored in local database, which is the knowledge base, to later empower callback process from the analysis tools. This callback will help association of model components in analysis process.

- **CR5: The trace model should support data classification by providing data levels of measurement**

  Specification: Additional metadata field to indicate data aggregation level `measurement_type`, which can contain any of the four measurement levels, *nominal*, *ordinal*, *interval*, or *ratio*. This metadata should be automatically programmed to the relevant data field.

  Example:

`susceptibility_se`: High

`data_domain`: Social data

`data_type`: Experiment data

`data_source`: Brighton experiment 16/09/2015

`created_by`: ethanitl

`last_updated_by`: ethanitl

`measurement_type`: **Ordinal**

- **CR6: The trace model should support control metadata of creation and last updated timestamps and status (Active, Inactive, Removed).**

  Specification: Additional non-modifiable metadata fields to indicate timestamp of creation `created_on` (format: DD/MM/YYYY) and `last_updated_on` to indicate timestamp of saved modifications made (format: DD/MM/YYYY). Higher granularity could be added by adding the exact hour (format: HH/MM/SS). The recorded timestamp is the timestamp when the user saves an ongoing model creation process. It is important that these two metadata fields are intertwined with the metadata fields `created_by` and `last_updated_by` to

[19] Ethan's username in the TRESPASS system.

answer both *who* and *when* questions. These metadata fields should be added to every data field that requires user input. Additionally, a `status` field that informs the validity of the component data as a whole should be added to every component, with three possible options to choose from: *Active* (the data is still valid and contributes to the analysis), *Inactive* (the data is valid but irrelevant to the analysis), or *Removed* (the data is no longer valid and used). The option *Removed* prevents the user from physically deleting obsolete data to allow future reference when needed.

Example:

`susceptibility_se`: High

`data_domain`: Social data

`data_type`: Experiment data

`data_source`: Brighton experiment 16/09/2015

`created_by`: ethanitl

`last_updated_by`: ethanitl

`measurement_type`: Ordinal

`created_on`: **27/10/2015 15:17:29**

`last_updated_on`: **30/11/2015 11:57:21**

`status`: **Active**

- **CR19: The trace model should be able to provide logger for the knowledge base that stores original data received and its associated metadata (timestamps, origins, version, etc).**

  Specification: For every data imported into the knowledge base, there should be automatically generated fields `timestamp` that indicates when the data was imported into the knowledge base, `origin` to allow the source of the data to be identified, and `version` to show the current version of the data as well to allow tracking should there be multiple versions of the same data expected.

  Example: Suppose that the user wishes to look up vulnerabilities that might be present in his computer. He seeks to find out whether his favourite browser, Firefox 20.0 and the subversion (SVN) software he regularly uses have potentially harmful vulnerabilities. To do this, he uses the tool Software Checker (available via `https://trespass.itrust.lu/tools`). The software works by means of installation in the machine to be checked. It supports XML formatted output. It turns out that the browser suffers from a number of *High* severity vulnerabilities, as reported by CVE (Common Vulnerabilities & Exposures), while the SVN software has relatively very few vulnerabilities. An excerpt of the finding is shown as follows.

```
<?xml version="1.0" encoding="UTF-8"?>
<Softwares>
<Software exist="true">
<name>firefox</name>
<version>20.0</version>
<Vulnerabilities>
<CVE CVE="CVE-2016-1946" severity="High"/>
<CVE CVE="CVE-2016-1935" severity="High"/>
<CVE CVE="CVE-2016-1931" severity="High"/>
<CVE CVE="CVE-2016-1930" severity="High"/>
<CVE CVE="CVE-2016-1521" severity="High"/>
.
.
.
</Vulnerabilities>
</Software>
<Software exist="true">
<name>subversion</name>
<version>1.6.23</version>
<Vulnerabilities>
<CVE CVE="CVE-2015-3187" severity="Medium"/>
<CVE CVE="CVE-2015-0251" severity="Medium"/>
<CVE CVE="CVE-2015-0248" severity="Medium"/>
</Vulnerabilities>
</Software>
</Softwares>
```

After the checking has been done, the user exports this finding to the knowledge base by in its XML format. Upon import, the following metadata are spawned:

`timestamp`: 30/11/2015 14:00:39

`origin`: Software Checker

`version`: 1

Similar to the fields `id` and `label`, the non-automatically generated additional metadata fields above ought to be set as mandatory fields to fill in whenever a user inserts a model component. One may regard this as an implementation of *techno-regulation*[20] [111]. By imposing such rules, we may move towards 'traceability by design' or ideally 'traceability by default'. The information provided for each model component should be associated only with the TRESPASS model that contains them, meaning that although the model elements can be obtained from component palettes or Model Pattern Library, the system should prohibit multiple usage of an `id` and `label`. The libraries/palettes should be configured such that they spawn fresh `id` everytime a new component is added. Similarly at a higher level, the system should prohibit multiple assignment of the same model ID, although this setting should already be configured (for example, by suggesting users to select *Save As* or to give warning for duplicating names). These data should be stored in the knowledge base along with the TRESPASS model.

Moreover, the system makes it possible for the user to fabricate models using pre-defined templates and generic model components from the available libraries, namely the Model Pattern Library (for model templates, such as a sample model for a cloud company) and Entity Type Library (for generic entities, such as Actors). This is linked to CR9 as described in Table 6.4, which can be decomposed to the following specification:

- **CR9: The trace model should be able to identify model elements derived from libraries (template and model elements from Model Pattern Library, attacker profiles from Attacker Profile Library, and annotated attack trees from Attack Pattern Library)**

  Specification: Each template/pattern from library used in the creation of the TRESPASS model must be easily identified, be it from the Model Pattern Library, Entity Type Library, or Attacker Profile Library. The metadata should be generated automatically upon import to the ANM and could be denoted by either of these fields: `templatemodel`, `genericattacker`, or `genericentity`, filled with *Yes*. The ANM should be able to mark and store this ID as a metadata in the TRESPASS model. In case the user does not make use of any pattern/template, this metadata should **not** be generated, making it easy to distinguish generic components from custom-made components.

  Example: Suppose to build the ITL model, the user utilises the pre-specified cloud company model in the Model Pattern Library. Besides the unique ID assigned to the TRESPASS model by the user, a metadata field `templatemodel` : *Yes* should be visible.

CR17 implies that the TRESPASS model should be able to generate unique IDs for data imported into the tool or library elements imported into the model. The specification for CR9 already encompasses the latter. The following specification concerns the first part of the criterion.

- **CR17: The trace model should be linked with the ANM as the main tool in creating the TRESPASS model and help generate IDs for data imported into the tool or library elements imported into the model.**

  Specification: Each data used in the components of the TRESPASS model must have a specific `data_ID`.

  Example:

`susceptibility_se`: High

`data_ID`: **susceptibility-defender-big**

`data_domain`: Social data

`data_type`: Experiment data

`data_source`: Brighton experiment 16/09/2015

`created_by`: ethanitl

`last_updated_by`: ethanitl

[20] Leenes defines techno-regulation as "deliberate employment of technology to regulate human behaviour".

`measurement_type`: Ordinal

`created_on`: 27/10/2015 15:17:29

`last_updated_on`: 30/11/2015 11:57:21

`status`: Active

Following the currently defined TRESPASS process, these metadata should be written in RDF statements (more information will be available in the forthcoming D2.4.1). These metadata will be stored in the TRESPASS Model File instance in the knowledge base and can be utilised by the knowledge base using the 'types' call. This is important because the TRESPASS Model File is required as input for the Tree Maker to generate the attack trees (using 'getData' calls). We later show that the metadata `data_ID` will be extensively used in the attack trees.

The user answers a number of general questions about the organisation in survey style (e.g. organisational culture). The answers are used to set properties of the items of the map, in particular the people/employees.

Information from this step can help fill in missing fields for the `actor` information, such as `personal_situation`. However, we skip this step as the information provided in the last step would already suffice to illustrate the rest of the Use Case.

The user inserts additional information regarding the model components in plaintext format (submenu 'Comments' or something similar). This information could be the context of the organisation when the model was created, the purpose of the analysis, important assumptions, components exclusion and inclusion decision, etc.

At this point, the TRESPASS model is more or less finished, but upon closer inspection there might be additional memos necessary to shed light on the context or other unwritten decisions leading to the finalisation of the model. Not all components in the TRESPASS model can be traced back to the tagged manual model. It might also be the case that clarifications need to be made at the components level, which demands extra explanation attached to the nodes and edges. The trace model should provide this feature, as outlined in CR10. CR10 can be specified further as follows:

- **CR10: The trace model should be integrated with the ANM as the central tool in the model creation stage and should support (1) insertion of generic comments for the model as a whole and (2) annotation of the model's nodes and edges for additional explanation.**

  Specification: Two additional features are needed to support this criterion. To enable the user to comment on the model as a whole, a model-wide sub-wizard is essential. This additional sub-wizard could be added to the ANM under the title **Comments** or something similar (see Figure 6.6). Under this sub-wizard there could be categories of comments that the user wishes to add, for example **Context**, **Purpose**, **Assumptions**, **Abstraction Level**, and **Others**. Under each subheading, a text field of arbitrary length is provided.

  The other essential feature is the ability to put comments into edges and nodes when needed, much like the ability to add comments to a specific part of a manuscript in .pdf format in Adobe Acrobat Reader®. To add comments, the user can simply right-click on the desired model nodes or edges, upon which a textbox will appear and user can type notes in.

  Example: In Figure 6.6 we show a superimposed screen capture of how the cloud model would look like in the ANM, originally presented in D6.3.1 to demonstrate the prototype of the interface. In this doctored screen capture, we add the additional sub-wizard **Comments** alongside other existing sub-wizards. We also show an example of a node-level comment on the model itself.

Again, the comments are stored in the TRESPASS Model File instance in the knowledge base.

The user answers a number of general questions about the threat environment. The user can select the attacker profiles of interest, and adapt these if necessary.

To incorporate attackers as Actors in the model, the user may choose one of these two avenues: choose from the available 22 pre-defined threat agent profiles based on Threat Agent Risk Assessment (TARA) [151] using the Attacker Profile Library and adjust the parameters as necessary, or manufacture a custom threat agent and add it to the library. Here, we suppose that the user wishes to model *Henry* (the replacement for Cleo) in the model (see Section

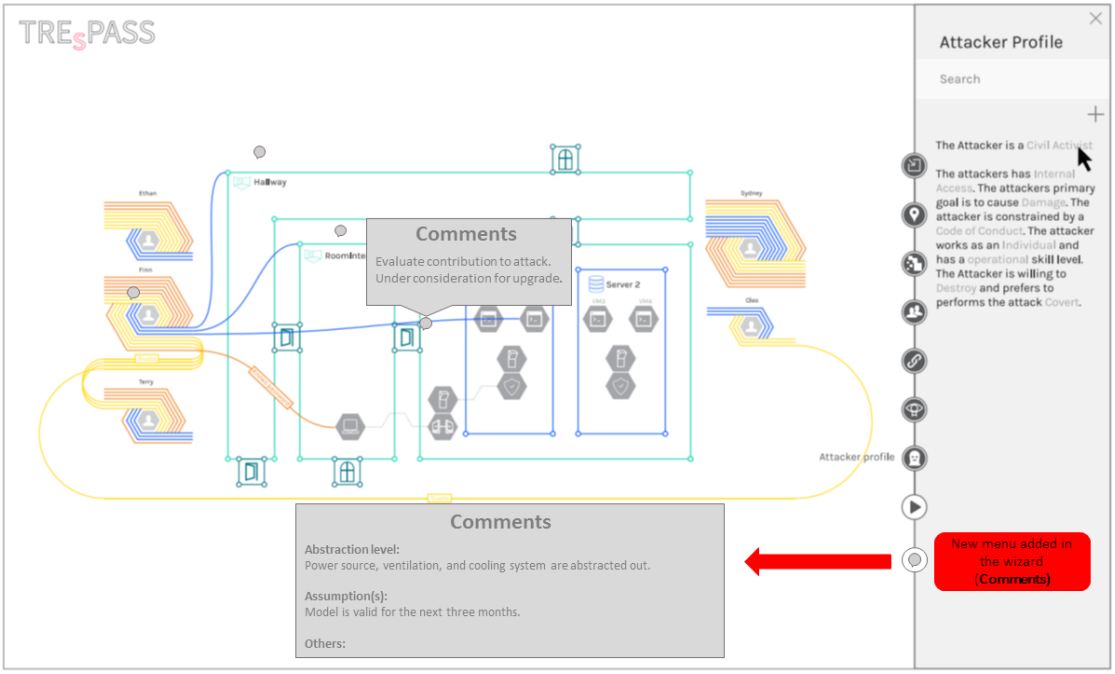


Figure 6.6: Sub-wizard **Comments** in Attack Navigator Map

source: modified from [196]

5.2). To do this, the user may select from the following plausible threat agent profiles: **Thief**[21] or **Vendor**[22] from the library to represent Henry.

This step correlates with CR18, in which we intend to have unique actor IDs specifically for attacker profiles to build linkage from the ANM to the Attacker Profile Library. Assuming that the profile **Thief** is selected, the specification for this criterion would look like the following:

- **CR18: The trace model should generate unique Actor IDs for attacker profiles imported into the ANM.**

  Specification: A parameter that denotes the unique ID for attacker should be defined. This is already contained in the current set of parameters for attackers (`id`).

  Example:

`actor`

`id`: **attacker-henry**

`label`: Henry

`skill_burglar`: None

`skill_se`: None

`time`: Days

`budget`: £50

`motivation`: Acquisition/Theft

`risk appetite`: High

`genericattacker`: Yes

[21] An "opportunistic individual with simple profit motive." [151]
[22] A "business partner who seeks inside information for financial advantage over competitors." [151]

The user starts the analysis and selects one or more of the following results (1) A list of possible attack scenarios, ordered by their risk; (2) A list of possible countermeasures, ordered by their cost-effectiveness; or (3) A visualisation of the map, showing 'the weakest links', for example by increasing their size, or changing their colour.

At this point the TREsPASS toolchain helps the user to transform the map into routes, showing possible attack paths that the attacker may select from in order to successfully reach the goal.

It goes without saying that the majority of the design criteria we have devised concerns the process behind the analysis stage. To better illustrate the processes that happen during this stage, we give a visualisation of Scenario 1 in the tool used to visualise attack trees, ADTool. We also annotate the likelihood of success for necessary steps. Attack-defence trees are a formalism that allows attack generation in TREsPASS [185]. The visualisation and the modelling language behind this attack-defence tree is given in Appendix G.[23]

We now begin to dissect criteria that are related to the analysis stage. In CR11, we formulate that each unique piece of data generated during the analysis process should have an unique identification. The listing shown above does not distinguish likelihoods between different attack steps (identified by `parameter domainId="ProbSucc1" category="basic"`, which corresponds to the domain **Probability of success** in the ADTool). To improve this, we propose the following specification for CR11:

- **CR11: The trace model should allow unique identification for each unique piece of data.**

  Specification: Each unique piece of augmentation and annotation data must have a different identification, be it likelihood, time, or cost. In practice, this is the task of APLTool that performs callback of data from the knowledge base. Having said this, we give an example using the APLTool.

  Example: Based on the APLTool demo available in `https://trespass.itrust.lu/tools`, we generate the following example using the provided input data for the IPTV case study. This does not bother our cloud case study that we have used so far, as the parameters shown (cost, difficulty, likelihood, and time) would also be the same for attack steps in the cloud case study. The output file is given in the following listing:

```
<node refinement="disjunctive">
<label>impersonate somebody trusted by Margrethe</label>

<node refinement="disjunctive">
<label>Impersonate carer</label>
<parameter name="cost-success-ratio" class="numeric">0</parameter>
<parameter name="cost" class="numeric">0</parameter>
<parameter name="difficulty" class="ordinal">M</parameter>
<parameter name="likelihood" class="ordinal">V</parameter>
<parameter name="time" class="ordinal">MT</parameter>
</node>

<node refinement="disjunctive"><label>Impersonate IPTV technician</label>
<parameter name="cost-success-ratio" class="numeric">142.86</parameter>
<parameter name="cost" class="numeric">50</parameter>
<parameter name="difficulty" class="ordinal">M</parameter>
<parameter name="likelihood" class="ordinal">M</parameter>
<parameter name="time" class="ordinal">MT</parameter>
</node>

</node>
```

  We propose to modify the identification of data by attaching the `label` of each node to its parameters, with the goal of having the parameters uniquely connected to their labels. Our proposal works under the assumption that label parsing is made available in APLTool, and therefore the labels embedded within the parameter names can be identified.

```
<node refinement="disjunctive">
<label>impersonate somebody trusted by Margrethe</label>

<node refinement="disjunctive">
<label>Impersonate carer</label>
<parameter name=\textbf{"cost-success-ratio_Impersonatecarer"} class="numeric">0</parameter>
<parameter name=\textbf{"cost_Impersonatecarer"} class="numeric">0</parameter>
```

[23] The TREsPASS socio-technical security modelling language (TPL) is explained in D1.3.3 [190]

```
<parameter name=\textbf{"difficulty_Impersonatecarer"}class="ordinal">M</parameter>
<parameter name=\textbf{"likelihood_Impersonatecarer"} class="ordinal">V</parameter>
<parameter name=\textbf{"time_Impersonatecarer"} class="ordinal">MT</parameter>
</node>

<node refinement="disjunctive"><label>Impersonate IPTV technician</label>
<parameter name="cost-success-ratio_ImpersonateIPTVtechnician" class="numeric">142.86</parameter>
<parameter name="cost_ImpersonateIPTVtechnician" class="numeric">50</parameter>
<parameter name="difficulty_ImpersonateIPTVtechnician" class="ordinal">M</parameter>
<parameter name="likelihood_ImpersonateIPTVtechnician" class="ordinal">M</parameter>
<parameter name="time_ImpersonateIPTVtechnician" class="ordinal">MT</parameter>
</node>

</node>
```

We could also establish a robust link from the APLTool to the knowledge base data. Currently the APLTool can be configured to perform 'callback' to the knowledge base (with the 'getAnnotations' call). CR20 seeks to establish this link by connecting data of Actors and Assets stored in the knowledge base to be called into the APLTool. We specify this criterion as follows:

- **CR20: The trace model should support leaf expansion in APLTool by linking Actors and Assets in the attack tree with knowledge base data. The trace model should also further be able to link these Actors and Assets with quantitative attack properties such as attack difficulty, skill level, or likelihood.**

  Specification: Part of this has already been exercised in the specification of CR11. We could enhance this by linking Margrethe, IPTV technician, and carer as Actors stored in the knowledge base.

  Example: In the knowledge base these actors would be declared as follows:

```
<actors>
<actor id="Margrethe">
<atLocations>"Unknown"</atLocations>
</actor>

<actor id="IPTV technician">
<atLocations>"Unknown"</atLocations>
</actor>

<actor id="carer">
<atLocations>"Unknown"</atLocations>
</actor>
</actors>
```

The Actors shown in the listing of CR11 can be connected to the knowledge base data by parsing the node labels in order to extract the `actor id` of the Actor from the declaration. This way, actors included in the label are recognised as specific elements that have already been included in the TRESPASS model.

```
<node refinement="disjunctive">
<label>impersonate somebody trusted by Margrethe</label>

<node refinement="disjunctive">
<label>Impersonate carer</label>
<parameter name=\textbf{"cost-success-ratio_Impersonatecarer"} class="numeric">0</parameter>
<parameter name=\textbf{"cost_Impersonatecarer"} class="numeric">0</parameter>
<parameter name=\textbf{"difficulty_Impersonatecarer"}class="ordinal">M</parameter>
<parameter name=\textbf{"likelihood_Impersonatecarer"} class="ordinal">V</parameter>
<parameter name=\textbf{"time_Impersonatecarer"} class="ordinal">MT</parameter>
</node>

<node refinement="disjunctive"><label>Impersonate IPTV technician</label>
<parameter name="cost-success-ratio_ImpersonateIPTVtechnician" class="numeric">142.86</parameter>
<parameter name="cost_ImpersonateIPTVtechnician" class="numeric">50</parameter>
<parameter name="difficulty_ImpersonateIPTVtechnician" class="ordinal">M</parameter>
<parameter name="likelihood_ImpersonateIPTVtechnician" class="ordinal">M</parameter>
<parameter name="time_ImpersonateIPTVtechnician" class="ordinal">MT</parameter>
</node>

</node>
```

To preserve consistency, we could also first assign a `node_ID` for each node, from which the parameter naming would follow. This is what we aim to achieve with CR22.

- **CR22: The trace model should be able to generate unique identification for attack patterns used in the annotated attack trees.**

  Specification: For every node generated in the tool, there should be a unique `node id`. This would happen in the APLTool. Naming could be done on an arbitrary basis to avoid confusion with numbers (`node001`, `node002`, etc.)

  Example:

```
<node refinement="disjunctive">
<label>impersonate somebody trusted by Margrethe</label>
<node id = "Impersonatetrusted">

<node refinement="disjunctive">
<label>Impersonate carer</label>
<node id = "Impersonatecarer">
<parameter name=\textbf{"cost-success-ratio_Impersonatecarer"} class="numeric">0</parameter>
<parameter name=\textbf{"cost_Impersonatecarer"} class="numeric">0</parameter>
<parameter name=\textbf{"difficulty_Impersonatecarer"}class="ordinal">M</parameter>
<parameter name=\textbf{"likelihood_Impersonatecarer"} class="ordinal">V</parameter>
<parameter name=\textbf{"time_Impersonatecarer"} class="ordinal">MT</parameter>
</node>

<node refinement="disjunctive"><label>Impersonate IPTV technician</label>
<node id = "ImpersonateIPTVtechnician">
<parameter name="cost-success-ratio_ImpersonateIPTVtechnician" class="numeric">142.86</parameter>
<parameter name="cost_ImpersonateIPTVtechnician" class="numeric">50</parameter>
<parameter name="difficulty_ImpersonateIPTVtechnician" class="ordinal">M</parameter>
<parameter name="likelihood_ImpersonateIPTVtechnician" class="ordinal">M</parameter>
<parameter name="time_ImpersonateIPTVtechnician" class="ordinal">MT</parameter>
</node>

</node>
```

This modification also contributes to the materialisation of CR23. Upon trying out the demo of Attack Tree Analyzer in `https://trespass.itrust.lu/tools`, we noticed that the nodes and the affiliated parameters previously defined in APLTool take on different forms of representation. An excerpt of the input file is shown below:

```
#Impersonate IPTV technician
X114 50 0.35 0 0 M MT

#Impersonate carer
X115 0 0.90 0 0 M MT
```

where `x114` would correspond to the attack node that we have previously defined as `ImpersonateIPTVtechnician`. It is also used in the representation of the formula used to analyse the attack tree. Furthermore, the parameters do not come with explanation (although the user could make an educated guess of what they mean based on the language used in APLTool). This inconsistency would cause confusion to the user and break the traceability chain. We propose to rectify this by devising a specification for CR23 as follows:

- **CR23: The trace model should enable linkage of quantitative properties resulting from the analysis of attack trees (e.g. attack difficulties, cost of attack, etc.) to the original attack trees generated with Tree Maker and visualised in ADTool.**

  Specification: The Attack Tree Analyzer should reuse `node id` and the affiliated `parameter name` defined in APLTool.

  Example:

```
# attack steps
.
.
.
#Impersonate IPTV technician
ImpersonateIPTVtechnician
cost_ImpersonateIPTVtechnician: 50
```

```
likelihood_ImpersonateIPTVtechnician: 0.35
dummyvariable1_ImpersonateIPTVtechnician: 0
dummyvariable2_ImpersonateIPTVtechnician: 0
difficulty_ImpersonateIPTVtechnician: M
time_ImpersonateIPTVtechnician: MT

#Impersonate carer
Impersonatecarer
cost_Impersonatecarer: 0
likelihood_Impersonatecarer: 0.90
dummyvariable1_Impersonatecarer: 0
dummyvariable2_Impersonatecarer: 0
difficulty_Impersonatecarer: M
time_Impersonatecarer: MT
```

Establishment of the `node id` aids the linkage between the visualisation of the results (navigator map) and the attack trees, which is what we attempt to design through CR24.

- **CR24: The trace model should be able to link nodes in the navigator map to the leaves of the attack tree as visualised in the ADTool.**

  Specification: When visualisation results are shown, the user can hover over a specific node to understand where it is coming from in terms of the attack tree. This is done by showing the node `id`.

On a related note, we also consider the storage of these parameters as formalised in CR12.

- **CR12: The trace model should have a data staging area that stores parameters into a local persistence data model.**

  Specification: To achieve CR12, the augmentation and annotation parameters that are generated by the APLTool stem from the knowledge base through 'getAnnotations'call. When the node-specific parameters have been generated, they are stored in the Attack Tree Augmented & Annotated instance in knowledge base.

The user selects the scenarios/attack trees to understand which components are linked to a specific scenario/attack tree. Upon clicking on a certain leaf node, the model shows model components that are involved, such as assets (annotated with their quantitative values), control mechanisms, and actors.

After the analysis process is finished, the user wants to deep dive into the corresponding results in order to understand how the analysis results can be understood in terms of the model components built earlier. In complex organisations with tens of actors and multiple layers of technologies involved, understanding such analysis results may take a substantial amount of time. Although TREsPASS allows generation of ranking of most plausible threats and attack paths, sometimes such system-generated ranking may seem counter-intuitive (especially to the C-suite) and therefore deep understanding is needed, at least to explain why the analysis results show attack paths *X*, *Y*, and *Z* are shown to be the most prominent and not *A*, *B*, *C* as the CEO had expected. Traceability towards (1) the model components involved and (2) the parameters attached will seamlessly help the user make sense of the analysis results.

We consider the need for the trace model to support such activities in CR15, CR16, and CR25. Dissection of these criteria into specifications is outlined as follows:

- **CR15: The trace model should support XML format as the main format used in TREsPASS model and attack trees. It should be able to identify Locations, Edges, Assets, Actors, Predicates, Policies, and Processes in the model.**

  Specification: Part of this is already demonstrated in CR11, in which we show traceability can be infused at the node-level in XML format. However, we also need to embed *components* in the language of the attack-defence tree.

  Example: Examples for how `<assets>` and `<actors>` can be embedded into the attack representation in XML will be presented using an excerpt of the attack-defence tree of Scenario 1.

```
<node refinement="conjunctive">
<label>Gain access to laptop</label>
<node refinement="conjunctive">
<label>Get temporary badge</label>
<node refinement="conjunctive">
<label>Spoof Big</label>
<parameter domainId="ProbSucc1" category="basic">0.35</parameter>
```

From the above excerpt we see that the laptop and the temporary badge are not identified as <assets>, and Big himself is not considered an <actor>. To link these components, naturally they need to be declared first (laptop and temporary badge in <assets> and Big in <actors>) in the TRESPASS model. The declaration would be shown as follows:

```
<locations>
        <location id="RoomInternal" domain = "physical"/>
        <location id="Outside" domain = "physical"/>
</locations>
<assets>
        <item name = "Laptop" id = "laptop-admin">
                <atLocations>"RoomInternal"</atLocations>
        <\item>
        <item name = "Temporary Badge" id="badge-temporary">
                <atLocations>"RoomInternal"</atLocations>
        </item>
</assets>
<actors>
        <actor id="defender-big">
                <atLocations>"Outside"</atLocations>
        </actor>
</actors>
```

Although this example was taken from an ADTool file that is not linked with any TRESPASS model, in reality the TRESPASS toolchain currently does not support such linkage. The reworked language would look like the following:

```
<node refinement="conjunctive">
<label>Gain access to laptop-admin</label>
<node id="laptopaccess">
<node refinement="conjunctive">
<label>Get badge-temporary</label>
<node id="gettemporarybadge">
<node refinement="conjunctive">
<label>Spoof defender-big</label>
<node id="spoofbig">
<parameter domainId="ProbSucc1" category="basic">0.35</parameter>
```

- **CR16: The trace model should be able to link Locations, Edges, Assets, Actors, Predicates, Policies, and Processes in the attack trees generation and analysis back to the model. The trace model should be able to connect visualised attack paths to the parameters generated during the analysis (difficulty, cost, time, probability).**

  Specification: The principle behind this criterion has already been demonstrated in the specification of CR15. However, it is equally important that this linkage can be explored at the user interface. We envisage this as the ability to click on the components in a leaf of an attack-defence tree and show the underlying (meta)data associated with them during analysis stage.

  Example:

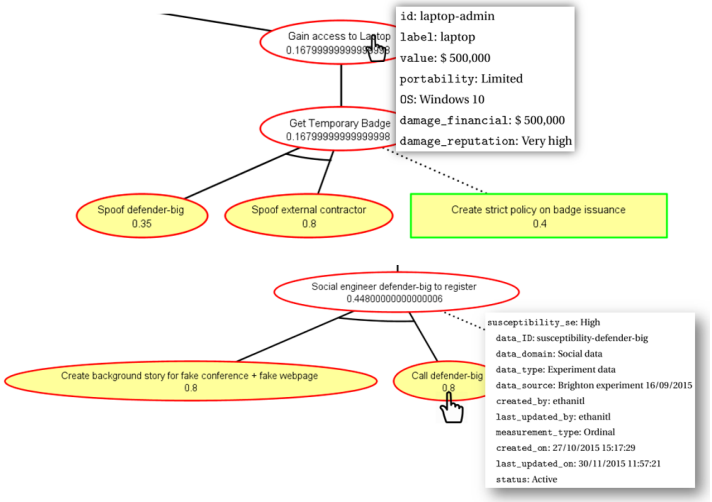
A beneficial feature that could help the user understand the analysis results in a holistic way is the ability to search and see how a specific entity/data shows up in the entire TRESPASS process. A prerequisite of this feature is to have an entity uniformly represented throughout the whole TRESPASS process. The specifications we have elicited thus far support such uniform representation, enabling seamless 'callback' of any piece of data specified as follows:

- **CR25: The trace model should enable callback of any piece of data by recalling its identifier and show where it shows up in the risk assessment process.**

  Specification: In the interface that shows the analysis results to user, a feature that allows global searching (similar to a search bar) is enabled. The user can type in a specific entity according to its unique ID and the result will show the locations in which the specific entity is used during the entire process.

  Example: Suppose that the user wants to look up where Mr.Big shows up in the process. He types in `id`: **defender-big** in the search bar, and the following result will show up:

  `id = "defender-big"` appears in:




```
location id = "Outside"
node id = "spoofbig"
```

(and other places in the attack-defence tree.)

The user annotates explanations of security decisions next to the analysis results in plaintext format. For example, the user can type in one or more security decisions that will be made in the next three months (or any time period) which corresponds to preventing a specific attack step/scenario. This explicitly shows couplings of decisions and the danger to be mitigated. Alternatively, the user can also provide a general list of possible countermeasures, among which the user selects a few to be applied within the next six months (or any time period).

After the user has arrived at a complete understanding of the analysis results and has communicated these results to relevant parties, the logical next step that would be taken by the organisation is to act upon these results. This trickles down to preventing possible attacks, translated into deciding upon appropriate security investments for a particular time period. As we have explained in Chapter 1, these decisions have to be carefully documented as future references might be necessary in case unexpected events occur, or simply for periodical control. A feature for this purpose must be assembled. The pertinent criteria for this purpose are CR26 and CR27, and their specifications are posed as follows.

- **CR26: The trace model should support plaintext format input in the navigator map in the ANM.**

  Specification: For this criterion, we envisage a very similar feature to the one specified for CR10. Within the user interface environment of the navigator map, the user is allowed to collate comments on a specific attack path, or compose a general comment for the whole navigator map. However, we do not yet have access to visualise how the final interface would look like, but it would look almost identical to the example for CR10. A textbox would appear upon clicking on a node, and a sub-wizard for general comments (**Comments**)

  Example: Suppose that based on the results, the organisation wants to focus on heightening the employees' overall awareness by deploying an awareness training program in the next 6 months. The rationale behind this is the high likelihood of social-engineering based attack paths. The user can type this in either separately at any attack paths that correspond with elements of social engineering attacks, or insert this in the **Comments** sub-wizard. Details may also be filled in, such as *next steps to be taken*, *project champion* to be held in charge for the deployment of the program, and *next cycle* to control the decision.[24]

- **CR27: The trace model should enable storage of the plaintext input in the navigator map in the ANM (in .txt format).**

  Specification: To enable easy reference, the collated comments should be stored in .txt format altogether with the visualisation results. Whenever the user opens the saved visualisation results, the comments will still be incorporated (both the node-level and the general comments).

[24]Requirements regarding reminders for scheduled analysis at certain intervals have been discussed in D6.2.2 [195].

Ideally these information should be stored altogether with the visualisation results. However, we do not have any information yet whether the visualisation results shall be treated as a separate instance - which would solve the storage problem were it true. Judging from the current development of the TRESPASS Information System, we propose to store these information along with the Analysis Results instances.

The model can be saved for future references or adjustments.
This step is self-explanatory.

6.3.2. Scenario 2 applied to Use Case 3

To articulate how support for traceability can also help in coping with organisational dynamics, we now demonstrate Scenario 2 (see Section 5.2.3) in a piecemeal approach to Use Case 3. The case will still revolve around ITL, only modified with the changes we have described in Scenario 2. The goal of this explanation is to devise specifications of criteria pertaining to the dynamic view of traceability, namely CR28 up to CR31.

The user receives push notifications from one of the available data extraction tools or requests automated data extraction. The push notifications inform the user that there have been updates in the data based on changes detected by the input tools (see Step 3).
The user receives a push notification from CVE via Software Checker, mentioning that a number of high severity vulnerabilities have surged. Remember that because the user has installed the Software Checker that is connected to the internal network, Software Checker can regularly verify the vulnerability state of the software used.[25]. Within the trace model, this is governed by CR28.

- **CR28: The trace model should be able to notify users based on push notifications of updates on external data sources or automated data extraction tools (facilitated through the knowledge base) and prompt the user whether to conduct a re-analysis or not.**

  Specification: All data-flowing tools connected to the knowledge base should be able to generate notifications on important updates, including the Attacker Profile Library (via ANM), the Entity Type Library (via ANM), online vulnerability databases (via Software Checker), or technical data (from virtualised environment data extraction tools). These notifications should not be bound to a regular interval, but rather spontaneous whenever new updates are available.

  Example:

The user opens the file containing the model used in the previous assessment. The file also contains the previous analysis results.
The user revisits the results of the most recent risk assessment activity (as presented in Use Case 1 and 2). The file contains the TRESPASS model as well as the analysis results and the visualisations. The user prepares to modify the TRESPASS model to represent changes that were informed to him automatically, and changes that happen in the physical realm and need to be translated into the model.

[25] more info is available at `https://www.itrust.lu/wp-content/uploads/2014/07/products_software_checker_flyer.pdf`

Following up on the push notifications, the user adapts the model where needed to represent the changes observed, by moving, deleting, or adding elements and changing properties.
The user adapts the model from the previous analysis. There are three major changes made, two of them directly concerning the model:

1. The newly unearthed vulnerabilities are incorporated into the knowledge base, although this is done automatically;

2. The value of ***fileX*** is changed to $ 600,000 (increased by 20%); and

3. A new actor is added to the organisation (Liam).

The updated data fields look as follows:
`file`
`id`: fileX
`label`: File X
`value`: **$ 600,000**
`size`: 250 MB
`damage_financial`: **$ 600,000**
`damage_reputation`: Very high

The user has to be aware that the `damage_financial` will also have to be changed to reflect this new asset value. Under `value`, the metadata should look as follows. The metadata for `damage_financial` will also reflect an update.
`value`: $ 600,000
`data_ID`: value-fileX
`data_domain`: Technical data
`data_type`: Asset value
`data_source`: Estimation from trading department
`created_by`: ethanitl
`last_updated_by`: ethanitl
`measurement_type`: Ratio
`created_on`: 27/10/2015 15:48:11
`last_updated_on`: 15/01/2016 09:42:43
`status`: Active

Finally, the user has to update every estimated value that is based on the the estimated value of ***fileX***, for example the `value` of the Laptop, as well as `damage_financial` of Sydney's account, the Laptop, and VM1. The next important update is the creation of new Actor, Liam.
`actor`
`id`: defender-liam
`label`: Liam
`susceptibility_se`: Low
`value_system`: N/A
`age`: 28
`gender`: Male
`nationality`: British
`damage_reputation`: Low
`personal_situation`: Unknown

The model of the organisation also changes based on the addition of Liam, which is presented in Appendix G.4. Although Liam is hired as an administrator, he is not yet granted full privilege to the cloud infrastructure during his first three months of training. Such notes that concern important modelling decisions can be inserted in the general comments sub-wizard (see Figure 6.6). The user can also insert a reminder to rerun the analysis once Liam is granted full access to the organisation's IT infrastructure.

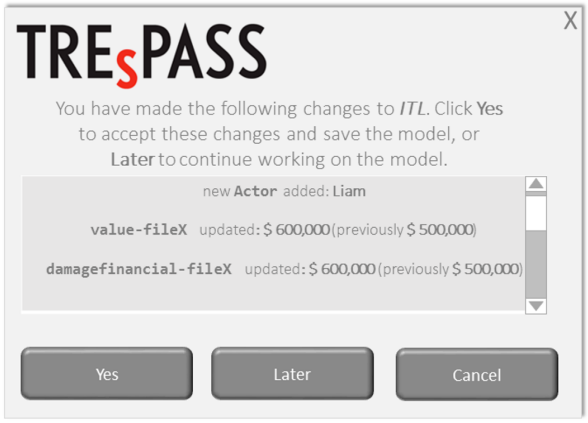
Alerts will be issued to the user when important updates have been made, or when user action is required to assist in the updates.

The user has altered the model according to the changes. He would like to save the new model, but the trace model has noticed some changes made to the model. This is a specification of CR29.

- **CR29: The trace model should be able to show differences across different model components in terms of TRESPASS concepts (e.g. Locations, Assets, Policies, etc.) and show the identifier of the different elements to enable easy comparison.**

  Specification: A notification to confirm changes made to the model appears when the save model command is triggered. The changes are shown in terms of their ID or Label to help easy identification. Previous values (if available) are also shown to aid easy comparison.

  Example:



The user re-runs the analysis to identify any differences in the analysis results given the new conditions.

If all changes are accepted by the user, then he can proceed by rerunning the analysis to compare how the changes affect the risk posture of the organisation. We attempt to achieve this through CR30.

- **CR30: The trace model should enable storage of analysis results and generate metadata that enables unique identification of each analysis result (e.g. timestamp, owner, version, etc.)**

  Specification: The analysis results from different models should be uniquely tied to their respective models. This could be done by means of metadata that establishes the timestamp (when the analysis was run and saved), ownership (who ran the analysis), or the version (which of the analyses is the most up-to-date). The easiest way to achieve this is to link the analysis results to the name of the file that contains the model, as two models cannot co-exist under the same name. Versioning works by the fact that the new analysis is based on an updated version of the old model.

  Example: Suppose that for consistency reasons, the previous analysis was named 'ITL November 2015' and the current analysis is named 'ITL January 2016'. To easily identify which analysis results belong to which model, metadata such as given as follows can be generated:

  `Filename`: ITL November 2015

  `timestamp`: 30/11/2015 16:37:22

  `owner`: ethanitl

  `version`: 1

  `Filename`: ITL January 2016

  `timestamp`: 15/01/2016 10:25:49

  `owner`: ethanitl

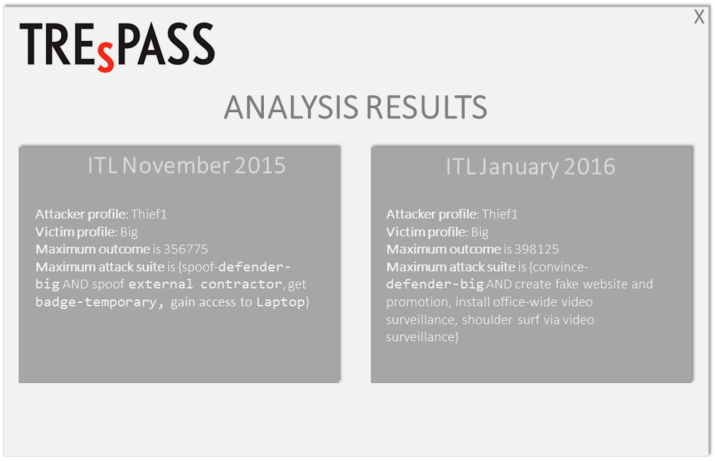
  `version`: 2

Alerts are issued when changes in the underlying data lead to significant changes in risk.
The user receives an alert mentioning that the changes generated a significant difference in risk. This is achieved through a side-by-side comparison and represented through CR31.

- **CR31: The trace model should be able to show the differences across analyses in terms of quantitative properties (e.g. difficulty, cost, time, likelihood).**

  Specification: Results from previous analysis and the current analysis can be shown side-by-side, enabling easy comparison. The user can select which files and what type of results he wishes to compare.

  Example:



The model can be saved for future references or adjustments.
This step is self-explanatory.

6.3.3. Summary of specifications

We have addressed design criteria that needed to be further extracted into structural specifications. These are criteria derived from functional requirements and user requirements. Design criteria of non-functional requirements (CR1, CR13, and CR14) have not been considered in the Use Cases, but will be given a small portion for discussion here. The specifications are summarised in Table 6.5.

Table 6.5: Summary of specifications

Criterion	Specification	Code
CR2	Metadata `data_domain` and `data_type` for model components.	SP2
CR3	Metadata `data_source`, `created_by`, and `last_updated_by` for model components.	SP3
CR4	Storage of model components metadata in knowledge base.	SP4
CR5	Metadata `measurement_type` for model components.	SP5
CR6	Metadata `created_on`, `last_updated_on`, and `status` for model components.	SP6
CR7	Functionality to import manual model during model creation.	SP7
CR8	Functionality to tag pictures during model creation.	SP8
CR9	Metadata `templatemodel`, `genericattacker`, or `genericentity` for generic elements or templates.	SP9
CR10	Sub-wizard **Comments** during model creation and functionality to add comments to model components.	SP10
CR11	Unique label for parameters in APLTool.	SP11
CR12	Parameters for augmentation and annotation are pushed back to the knowledge back for storage.	SP12
CR15	Linkage of model components in XML format to attack tree nodes.	SP15
CR16	Functionality to show metadata fields of model components in attack tree nodes.	SP16
CR17	Metadata `data_id` for model components.	SP17
CR18	Metadata `id` for attacker profiles.	SP18
CR19	Metadata `timestamp`, `origin`, and `version` for data imported into the knowledge base.	SP19
CR20	Linkage of Actors or Assets to unique parameter labels in APLTool.	SP20
CR22	Metadata `node id` for nodes generated in APLTool.	SP22
CR23	Linkage of `parameter name` defined in APLTool to analysis tools.	SP23
CR24	Functionality to show `node id` in visualisation results.	SP24
CR25	Functionality for global search of entity.	SP25
CR26	Sub-wizard **Comments** in visualisation interface for general comments and functionality to attach comments to individual nodes visualised.	SP26
CR27	Storage of comments in navigator map in .txt format.	SP27
CR28	Push notifications of updates from external data sources or automated data extraction tools.	SP28
CR29	Functionality to generate summary of changes made to the model.	SP29
CR30	Metadata `timestamp`, `owner`, and `version` for analysis results.	SP30
CR31	Functionality for comparison of different analysis results file.	SP31

CR1 and CR13 concern the implementation of data management process in TRESPASS for the trace model, which is a principle in itself within Work Package 2 on data management [191]. The data management process essentially consists of two main processes: data transformation and data manipulation [191]. Within data transformation, two

activities exist: data mapping and data transformation. We proposed to connect different data schemas through creation of several metadata fields, as shown in the specification for CR2, CR3, CR5, CR9, and CR17. Such metadata fields, created early during model creation phase, will help form seamless connection between processes. For data manipulation, we have proposed to support data input module by providing logger (CR6 and CR19) whenever new data are being stored. Moreover, we also tried to build data query capability through the trace model (CR25). In sum, the core of the specifications mainly revolves around TRE$_S$PASS data management process. With regards to CR14, we have repeatedly shown through the examples that the trace model works on the basis of the TRE$_S$PASS base model. Citing D2.1.2, the base model is required to represent *location*, *actor*, *behaviour data*, and *social vulnerabilities* [191]. Different examples of the specification have shown how the trace model works by utilising TRE$_S$PASS concepts instead of creating new concepts, by tinkering with the metadata behind them.

To evaluate, we acknowledge a few implementation challenges in substantiating the specifications, including:

- Risk assessment process will take more time due to completion of metadata fields;
- The file size of the results could be large due to all attachments and added metadata;
- Limitation of the artificial language (XML). It might be the case that the adaptation we propose are not implementable; and
- Compatibility between tools, albeit the common language they share.

To tackle the first challenge, we suggest automating the completion of the metadata fields whenever possible. The second challenge seems to be inevitable. A trade-off between compactness and complete traceability is present, and it is up to the user to establish the granularity of the traceability. He could, for instance, disregard attachments of manual modelling to save space. The third challenge could be combated by reconfiguring the data XML schema that has been defined as deemed necessary. Some elements (elements, attributes, types, or groups) might need reworking to accommodate a valid structure, valid data content, and default relationships between types of XML data [191]. Finally, compatibility between tools are not inherently a traceability problem, but the effort to constitute traceability might have impacts on tool compatibility. Extra attention must be devoted to solving such potential issues, especially during tool integration tasks.

6.4. Evaluation

In this section, we evaluate the design process as a whole, starting with how we process the requirements all the way to arriving at specifications. In whole, this includes a separate test of sufficiency for:

1. The goals;
2. The requirements, the assumptions, and the specifications; and
3. The relationship between (1) and (2).

Point (1) has been covered in Section 5.1.4, and parts of point (2) in Section 6.1.6. This section then extends the evaluation to include the specifications and point (3). According to Verschuren and Hartog, evaluating specifications take three dimensions: *plan* (evaluating possible structural alternatives), *process* (methodological check of specifications), and *product* (checking compliance with design criteria and the adequacy of the specifications to feed forward subsequent stages) [213]. We begin by looking at structural alternatives. We believe that there are no better structural alternatives, because what we have proposed are based on currently available processes and best practices in TRE$_S$PASS. Whenever possible, we align these structural specifications with actual requirements already devised in TRE$_S$PASS within various work packages. A complete structural alternative - for example one using a totally different modelling language - would demand a complete redesign of the process, and would very likely render three and half years worth of progress useless. In other words, we considered 'infrastructural circumstances' [213] by making our proposed specifications adjustable to current TRE$_S$PASS processes. We believe such choices would not receive much resistance from stakeholders of the project, as opposed to a completely 'destructive' structural alternative.

Moving on to process evaluation of the specifications, we now look at how we have arrived at the specifications. As demonstrated with the Use Cases, we formulated the specifications based on possible interaction between the user and the system to achieve a certain goal. This means that the specifications that we have created will not exist in vain. A point we have repeatedly alluded to is that traceability is a support for the activities carried out using TRE$_S$PASS. With this end in mind, the specifications we have outlined exist for one reason: to support the user with achieving their goal of using TRE$_S$PASS. More importantly, we also align this with the navigation metaphor and the

service model. This ultimately implies that the traceability support would not disrupt interactions between a client and the consultants but would render them more meaningful.

Finally, the hierarchical cascade from design criteria to structural specifications guarantees that the specifications comply with the criteria they are attached to. This way, we reduced the risk of committing both under-specifying and over-specifying. Naturally non-functional requirements were not specified further. Although we do not consider building a prototype for this thesis, we believe the examples we provided for the specifications can provide enough guidelines were they to be translated into a tool support.

Now we peruse the relationship between (1) and (2) by assessing the *fitness* of these aspects. As (1) and (2) should form a goal-sub goal relationship, point (3) can then be evaluated by referring to three facets [213]:

1. Are lower demands in the hierarchy adequate to achieve the next higher demand?;
2. Do lower demands achieve more than intended?; and
3. Are the sub goals logically consistent with the goals?

We believe all three aspects are already satisfied. Our approach of closely decomposing the 'higher' demands prevents the lower demands from sidetracking and achieving 'non-goals'. Effectiveness is achieved by having one specification for each criterion, and one criterion for each requirement. This way, we ensure that implementation of a specification will ultimately contribute to the requirement linked to it. Given adequate and suitable resources, translating these relationship into a prototype is a feasible option. However, we are constrained with the closely approaching project deadline which prevents us from doing so.

The relationship between goals, requirements, criteria, and specifications is summarised in Figure 6.7. In Chapter 7 we shall investigate whether the requirements, criteria, and specifications can satisfy the goals, through a single-case mechanism experiment.

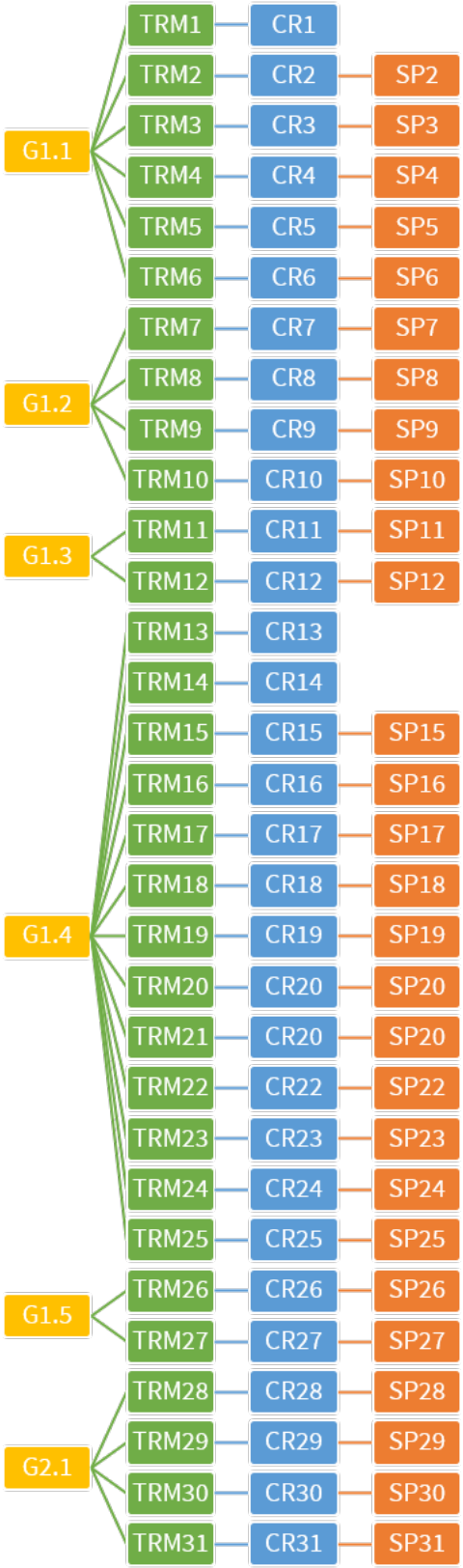


Figure 6.7: Relationship between goals, requirements, design criteria, and structural specifications

7

Treatment validation

This chapter closes the design science cycle we have started in Chapter 5. The main aim of this chapter is to validate the trace model that has been designed, with two goals in mind: to see whether the trace model has fulfilled its every requirement and to study its behaviour in a different context from the one in which it was designed. As explained earlier in Chapter 1, the term validation refers to both verification of requirements satisfaction and testing in a different context. We first present the results from the verification sessions with the experts and then explain the case study for the testing, followed by the results of the single-case mechanism experiment according to the protocol. We conclude the chapter by listing the validated specifications based on verification with experts and single-case mechanism experiment.

7.1. Expert opinion

As mentioned in Section 3.2.3, the focus of verification with experts is drawn towards evaluating the opportunities for implementation and whether the trace model successfully fulfilled its requirements. To achieve this, we contacted the experts that were involved in the semi-structured interviews and set up individual verification sessions. During the sessions, we presented the current design and exchanged views on how it currently stands. Notwithstanding the fact that not all experts were available for such sessions, we ensured that each requirement and design criterion received attention. Each verification session is unique; we focused on different requirements based on the expert's concerns and suggestions during the semi-structured interview. We also consider their expertise and their respective roles within the project.

The discussions were centred around practicality issues, namely how the design criteria could be implemented (given adequate resources) and how the current technical specifications (e.g. language of the tools, data storage arrangements) could support implementation (and what necessary adjustments would need to be made). In total, eight experts participated in the verification sessions, namely:

- **Christian Probst** (Technical University of Denmark/DTU);
- **Axel Tanner** (IBM Research Zürich/IBM);
- **Frederic Brodbeck** (Lust/LUST);
- **Wolter Pieters** (Delft University of Technology/TUD);
- **Margaret Ford** (Consult Hyperion/CHYP);
- **Lorena Montoya** (University of Twente/UT);
- **Dan Ionita** (UT); and
- **Olga Gadyatskaya** (University of Luxembourg/UL).

Some of the discussion touched upon very technical details, such as how the information system is built, how the logic for the knowledge base is ordered, and other issues. We also learned that some of our specifications have actually been implemented. Nevertheless, discussions with a bird eye's view on the overall process also existed.

This provides us balanced inputs on the feasibility of the implementation for the trace model. As is the case with semi-structured interviews, transcripts are not included for confidentiality reasons. In the subsequent sections, we distinguish the opinions for specifications of static view goals requirements (G1.1 up to G.1.5) and the specifications of dynamic view goal requirements (G2.1) for easy comprehension. Because the discussion revolves a lot around TREsPASS Information System, we advise the reader to start with an overview of the TREsPASS Information System, shown in Figure 7.1.

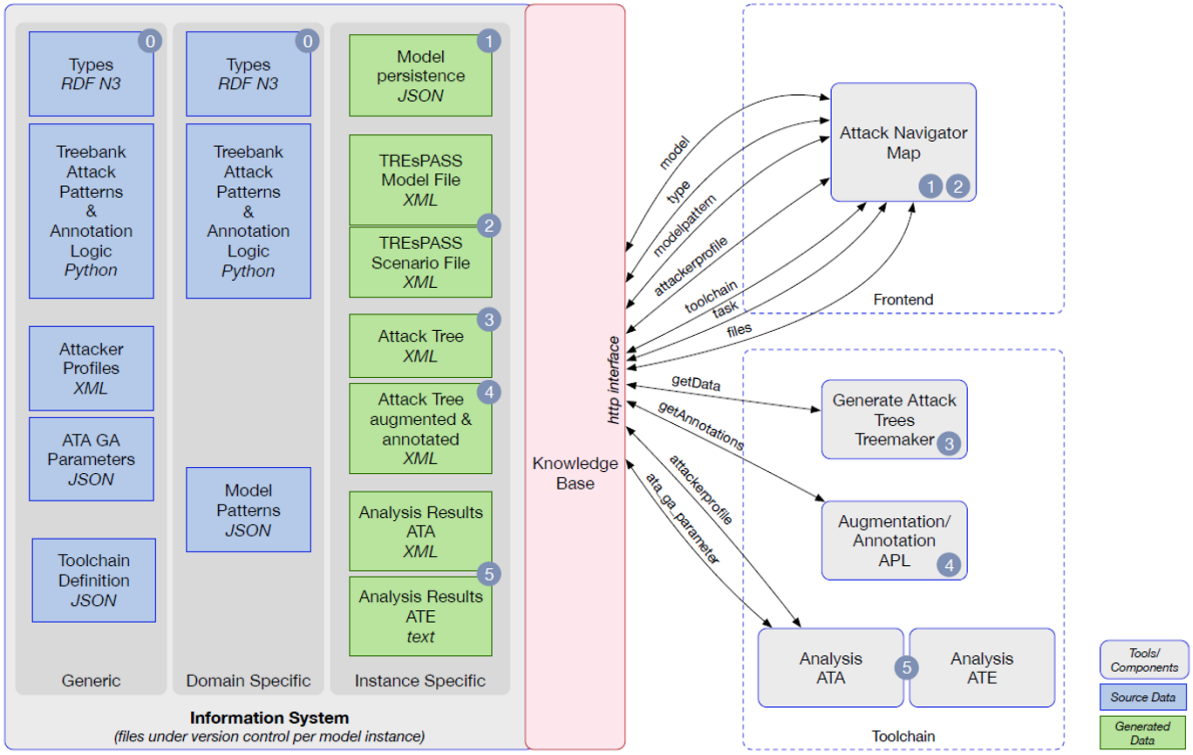


Figure 7.1: Overview of the TREsPASS Information System

Source: [200]

7.1.1. Opinions on the specifications of static view goals requirements

G1.1

We begin by zooming into the specifications listed under G1.1. This goal states that the trace model should enable users to trace data used in analysis back to the respective data sources. During the verification sessions, the specifications on metadata (SP2, SP3, SP4, SP5, and SP6) were grouped under one cluster. An expert especially saw this as a worthwhile feature to register expert opinions, which is one of the main sources of social data in TREsPASS . There were also suggestions to provide additional metadata for recording the date of data collection activity (for SP3) and also data aggregation level (SP5) to indicate the level of granularity within which the data was collected. According to experts of the information system, these metadata entries can be inserted as N3 types which allows arbitrary input. Configuration of data types in the knowledge base essentially allows multiple files for call operations, allowing as many input files as needed. If needed, the social engineering probability could be separated in an independent file (not in a domain specific file as currently set up). The parameter `susceptibility_se` shown in the example in Chapter 6 exists under the name `soc_eng_probability` in the knowledge base. A text editor interface would have to be created for metadata fields that require manual input, and it was also advised that they come with necessary background information and explanations of the possible values (e.g. `measurement_type`, with explanation on different types of scales.)

G1.2

G1.2 requires that model components included in the TREsPASS model can be traced back to the modelling decisions, which often occur manually. SP7 and SP8 both deal with the ability to upload the manual model leading to the

TRESPASS model and annotate them with the ID of the model components. The experts agreed that this is a potentially beneficial feature which could be supported by an intuitive user interface and API calls. They were also leaning towards the possibility of having the model components declared through the tagging activity, rather than doing an ex-ante approach of first building the TRESPASS model then returning to the attachment for tagging (cf. Section 6.3.1). The implementability is however yet to be explored (e.g. how to connect the tagging process with the model building process). An expert inquired the possibility of uploading the meeting notes (without the need for tagging them), which we think would be possible. The TRESPASS Model File instance is seen as a fitting location to store these attachments and tags.

SP9 aims to identify generic model components (library templates) explicitly. It was advised that instead of only providing *Yes* as a value, two more values should exist: *No* to identify custom-made components added by the user, and *Modified* to distinguish customised template components. Similar to the metadata specifications in G1.1, the metadata to identify generic model templates/components could be stored in another N3 file (that stores all libraries' data) and called during the data type call. Finally, SP10 mandates that users should be able to append both node-level comments and generic comments to the TRESPASS model. A concern was raised with respect to the decision to provide free text input versus a structured input with specified headings. The rationale to opt for more structured input format (especially in the general comments) would depend on how often the user would examine the comments. There is risk of overusing free text, although it is argued that this would be more usable from the user's perspective. Implementation of this specification should focus on developing the user interface (e.g. creation of the sub-wizard) and back-end call to knowledge base. An expert argued that this should be an optional thing instead of a mandatory step before running the analysis, to allow a quick scan without having to provide extensive comments.

A noteworthy observation is that differing opinions were raised with regards to the storage location of the comments. Some experts argued that conceptually the comments should be persisted in the same set of files in which the model components are stored. For example, a comment on an actor should be stored in the N3 file of the parameters. Although theoretically both types of commments can be stored in the TRESPASS Model File instance, the parties responsible for maintaining the knowledge base would prefer not to flood the TRESPASS Model File instance with information that do not contribute to generation of the attack tree (such as these comments). A solution to this would be to devise a new container for storage purposes.

G1.3

G1.3 refers to the ability to pinpoint data throughout the TRESPASS process, which is decomposed into SP11. There have been different opinions regarding how SP11 should be substantiated. One opinion voiced the need to connect the parameters within an annotated attack tree node with its label: similar to how we specify SP11. However, a suggestion to improve the label format was given by proposing not to have strings as labels to avoid conflicting names, but to use a structured alphanumeric format (e.g. `APL001` for APLTool). Consequently its parameters would look like `cost_APL001`, `difficulty_APL001` and so forth. This could be achieved by creating templates to assign IDs with the help of XSLT (eXtensible Stylesheet Language Transformations) to transform the XML files. Another opinion suggested a form of complex searching mechanism and not complicate the indexing of the parameters, for the reason that the parameters are already implicitly contained in a specific node. This could be achieved with the help of search and index tool software Lucene.

G1.4

The bulk of the verification sessions was focused at specifications related to G1.4, which seeks to enable user to trace analysis results basically to the steps that precede the analysis. With SP12, we intend to regulate the storage the intermediate parameters that are generated during the annotation and augmentation by pushing back to the XML file of Attack Tree augmented & annotated. It turned out that the current information system already exercises this. These parameters are stored because the ANM will need to call them later on for analysis. Besides, they are also checked into the Git system. In SP15, we propose embedding of model components in attack tree labels. Again, it turned out that the Tree Maker already implements this embedding in construction of the labels of attack tree nodes. These labels would stay intact until the annotation process in APLTool, but not in augmentation process, where different nodes will be combined to form attack steps. This automatically also reasserts SP20. However, care must be paid to ensure labels of the nodes in subleaves still contain the correct IDs of model components. Identification of attack patterns is addressed in SP22. We propose to add to a node ID to the leaves of the attack tree. This is argued as a redundancy because the node labels are already considered to be unique enough to act as an identifier to the node.

SP16 concerns the ability to view parameters of a specific model component when it is visually shown in an attack tree through interaction with a node's label. It is however hard to achieve this as currently the standard TRESPASS

process does not show visualised attack trees to the user, at least until the visualisation stage arrives. During the analysis stage, the attack tree can potentially be visualised via the Java-based ADTool before it gets annotated and augmented by APLTool. Instead, some experts vote that this should be done in ANM during visualisation stage. ANM has an advantage in that is integrated with the knowledge base. This is discussed in SP24, where users can view the ID of each node in the visualisation results (albeit arguments that labels are already unique enough to serve as identifiers). The labels are designed such that their structure is retained until the visualisation stage. Care must be taken to provide an interface that doesn't add complication to the visualisation, such that it requires deliberate user action to display the parameters (hovering/clicking), as the visualisation results could be "very busy in itself"[1]. Further development of this feature could involve link to the quantitative properties attached to a node (e.g. showing the calculation/expansion operation in APL that leads to the likelihood value in a node).

SP17 concerns the assignment of unique identifier to every specific data instance in the TREsPASS model. However, our assumption that such specific data instance exists is untrue: most parameters are described in classes with predetermined instance-specific values. It then makes little sense to have a very specific, instance-level identifier when uniqueness only exists at class level. If this specification is to be vetted, then the label of the component containing the parameter data plus the ID of the component would already suffice to construct a unique identifier. Another specification that proposes unique identifier is SP18, meant for attacker profiles. Such unique identifier apparently already exists at multiple levels. The standard process in ANM generates unique node ID for every model component dragged into the map (e.g. `id-r1mzEPE7-node`). Actors have their own identification under the format of `actor_class`, plus their labels which are assumed to be unique. Finally, attacker profiles have their own identifier (`APxxxx`). To add another identifier would be very voluminous and the user can make use of one of these identifiers.

Data timestamp, origin, and version history are proposed through SP19. The knowledge base currently already supports automated generation of timestamps for imported data, for example cloud-specific vulnerabilities (which is kept in the domain-specific Treebank Attack Patterns & Annotation Logic instance). Data origin is not explicitly stated, but it could be identified by looking at the type of data contained in the file. In the case of vulnerabilities, this is rather straightforward as the vulnerabilities have a prefix of `CVE-` to indicate the source. However, it was not revealed during the verification sessions how the origin of other data sources can be identified, for example from binary technical data. Versioning is also not explicitly stated (and occasionally argued as less relevant than the other metadata), but can be explored as every data stuffed into the knowledge base is checked into a Git system which implictly keeps track of version history. Other valuable suggestions include making explicit the command line or the instructions needed to generate the same type of data in the future in the Python scripts and adding the metadata that denotes the permission to share specific data (Restricted/Public).

SP23 discusses the possibility to modify the parameters of an attack step in Attack Tree Analyzer (ATA). It was revealed that the input language has been transmuted into standard XML, which provides more clarity in understanding the values used in the calculation (an example of this is shown in Appendix G.5.2. A functionality of a search bar to enable intelligent searching of model components is proposed in SP25. Experts see this as a feasible and beneficial feature when coupled with a powerful visualisation of the search result, for instance a split-screen view of the model and visualisation results, with capabilities to highlight the components that are being looked up. Another suggestion is to give a 'clean' interface by showing only relevant visualisation results and filter out the rest. The Lucene-assisted complex searching previously hinted at could be employed in this search bar. Although actors can move between locations, these movements are not manifested in the attack tree. Therefore, an actor would remain in the original location in the TREsPASS model and not show up in more than one locations when being searched.

G1.5

Ultimately G1.5 aims to increase accountability in the risk assessment process by enabling users to affix security investment decisions based on the analysis results. The feasibility to attach comments to multiple nodes in an attack path will have to be evaluated (e.g. with regards to how would the underlying language for the comments look like), but a feature for general comment text field and a node-specific text field is plausible. Possible uses of this feature may include reference to known vulnerabilities that are affecting the system to provide a more solid rationale as to the selection of the countermeasure and context explication: that at the moment the analysis was run only these vulnerabilities were known and not the others. The user can also include the person who would be responsible for the implementation and the next period to control the implementation of the countermeasures. It could be the case that a fresh instance would have to be generated to store these comments. Experts however emphasised the

[1]For samples of visualisation results, the reader is invited to explore D4.1.2, publicly available in `http://trespass-project.eu/sites/default/files/Deliverables/D4_1_2_0.pdf`.

importance of a powerful user interface so that users can fully leverage this feature. It was also advised to go for HTML or RTF format (instead of plain text) for the comments to allow formatting for users (for example, adding bullet points or giving emphasis in bold typeface)

7.1.2. Opinions on the specifications of dynamic view goal requirements

The dynamic view goal is captured in G2.1, which ideally would equip TRESPASS with the ability to propagate changes in the input to the analysis process correctly. Differing opinions surfaced during the discussion. We start with SP28, which seeks to identify changes in the input data and provide push notification in order for a re-analysis to be done. Some experts argued that this would be too difficult to implement given the current status of the TRESPASS project. Automated update capabilities are not yet supported and manual requests of data update would still have to be made. However, this is theoretically achievable by programming the ANM and knowledge base, for example in the case of virtualised infrastructure data in which latest updates of a network configuration would be imported automatically. With vulnerabilities, it was mentioned that it is possible to write an automated script that would command the Software Checker to periodically run and possibly notify the user for vulnerabilities whose severity reaches a certain threshold. Still, the bulk of the work if this were to be implemented remains in designing the GUI that would conveniently notify the user.

With SP29 we propose a notification of a summary of changes made to a specific TRESPASS model file. Implicitly this information is available and saved in the background, thanks to the Git system which allows generation of differences (diff)[2]. The capability to commit changes to the Git system is an important enabler of this feature. However, translating the differences into an attractive pop-up menu is a different story that involves further user interface development, possibly in the form of a list of changes as we envisaged. It was also suggested to enable the system to highlight nodes in the attack tree that would be affected by changes in the model before the changes are registered into the system; somewhat more directed towards a sensitivity analysis. However, this is currently not supported by the system as the Git can only compute diff when the changes is committed into the system. A less ambitious suggestion given is to at least present a warning to the user that changes have been made to the model (without specifying the changes) and hence the analysis result is invalid, requiring the user to run another analysis.

The Git system proves to be useful again if SP30 were to be implemented. Timestamp is produced in every version updates and version control is conveniently kept. A caveat could be that it might not be possible to have file owner updated as the Git does not recognise difference in user names, although an expert has vetted this feature to support accountability for changes. It was posited that a metadata that keeps track of the tools used in the toolchain would be worthwhile, although currently the Git system does not retain this information. Eventually, implementation of SP31 that shows comparison between analysis results is seen as very complicated. This is firstly due to the different analysis results that can be generated and the calculation of some metrics is still shaky (for example maximum attacker outcome). Secondly, no such mechanisms to reach this currently exist, which implies a call for a new tool. There were also varying opinions regarding TRESPASS ' capability of generating a rank of most likely attacks: some believe this feature exists but some doubt that the project has a clear decision on this feature. Were such comparisons of analyses made possible, some experts have expressed their wish to see comparison of top countermeasures and their associated costs, and comparison of visualisation results as well.

As a concluding remark, the general comment with regards to the implementation of the specification was positive. With sufficient time and resources, the experts believe that most of the specifications can be implemented, unless when stated that it is very difficult given the current status of the project or when there are past decisions that force us to be path-dependent. When asked to prioritise specifications for implementation, varying answers came up. Some believed that prioritisation is unlikely as the specifications depend on one another, but some also came up with ranks as to which specifications to prioritise, for example the ability to generate unique identifier and the mapping between analysis results and the attack tree plus the model components being placed high in the list. We found the verification sessions to be particularly insightful in checking whether our design is aligned with the current TRESPASS process and whether the two have overlaps. Time was an apparent limitation during the verification sessions (we set a limit of one hour per session), which disallowed us to extensively discuss every requirement with each expert. Our decision to only verify requirements based on the concerns of each expert might have also introduced bias during the verification sessions and restricted the variety of inputs. To avoid this bias, we ensured that each requirement was adequately addressed by different experts in different verification sessions so that multiple perspectives were considered.

[2] `https://git-scm.com/docs/git-diff`

7.2. Single-case mechanism experiment

The case study for the single-case mechanism experiment is adopted from Ionita et al [91], which essentially depicts a generic IaaS infrastructure shown in Figure 7.2. It is originally used to explore the possibility of conducting an argumentation-based risk assessment. The infrastructure consists of a physical layer owned by the Cloud Provider and a virtual layer owned by the Cloud Customer (in this case, ITL). The physical layer has several physical servers. A Hypervisor is running on top of each server, on top of which several virtual machines are running. A Cloud Administrator manages the whole physical infrastructure. The Administrator has access to the virtual machines (that belong to the customer). The virtual layer consists of a number of virtual machines, networks, and databases. Every cloud customer has control over a subset of virtual machines based on the server it utilises. For each cloud customer, a customer administrator is responsible for the configuration and management of virtual machines. All four virtual machines (VM1, VM2, VM3, and VM4) are now located in one virtual server and are connected via two virtual switches (VSW1 connects VM1 to VM2, and VSW2 connects VM3 to VM4). A virtual firewall (VFW) safeguards connections from the Internet. ***fileX*** still resides in VM1.

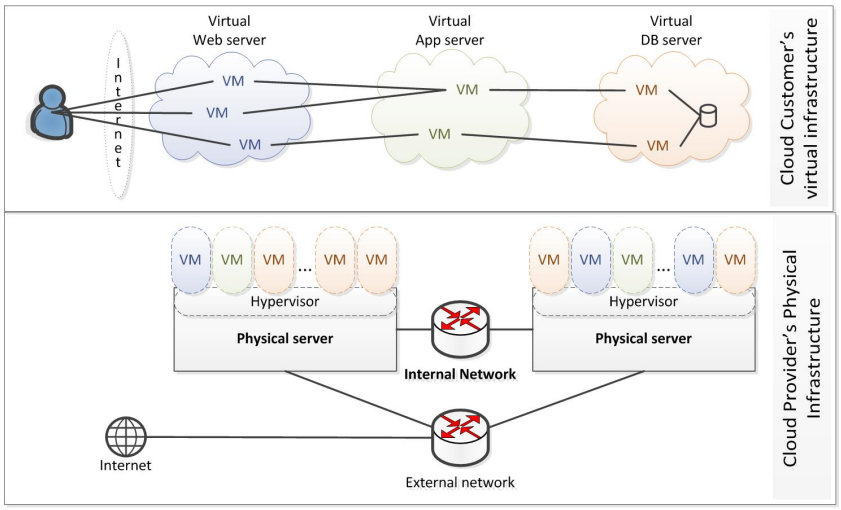


Figure 7.2: Cloud architecture for the test case

Source: [91]

Behind the selection of this case for the testing lie a few fundamental reasons besides the methodological consideration (cf. Appendix D). Firstly, the case has an outsourced cloud setting and assumes that ITL is utilising a public cloud. Consequently, it introduces a new entity into the picture, namely the *cloud provider*. In reality this is a very common situation, in which a company shares its cloud resources with other cloud tenants in a public cloud. We believe that providing a fundamentally different context allows for a more interesting analysis and deeper evaluation of the trace model in a different setting. This way, suggestions for improvements can be formulated in a more generic approach when we consider two entirely different contexts. The second reason is practicality: the case is publicly available and is presented in a simple manner, allowing room for adjustments for the testing purpose.

To make the example more relevant to our situation, some assumptions and adjustments are made, namely:

- The cloud infrastructure is a public cloud offered by a third-party provider. ITL is sharing resources with other cloud customers. The number of co-residing customers is unbeknownst to ITL: a realistic situation that also happens in real life;
- Focus is drawn towards the virtual DB (database) server that ITL puts in the cloud. Virtual Web server and virtual App server are abstracted out; and
- The room *Data Center* no longer exists in ITL's office, as they have migrated the physical IT infrastructure into a virtual IT infrastructure. As an IT administrator, Sydney still has full access to ITL's entire virtual infrastructure through his Laptop.

Similar to Chapter 6, we will employ Use Case 1 and 2 to guide the testing process of the static view goals, and Use Case 3 for the dynamic view goals. ITL's situation for Use Case 1 and 2 is the same as demonstrated in Chapter

6 (i.e. changes in Scenario 2 in Chapter 5 are not considered). In the demonstration of Use Case 3, different changes are formulated. The subsequent section presents a description of the scenarios.

7.3. Use Cases and Scenarios

We first demonstrate Use Case 1 and 2 with Scenario 1, followed by Scenario 2 applied to Use Case 3. Explanation of each step of the Use Case will not be as thorough as Chapter 6. Focus will be drawn towards processes/components that might differ from the context in which the trace model was originally designed.

7.3.1. Scenario 1 exercised in Use Case 1 and 2

The scenario for Use Case 1 and 2 is loosely adapted from the argumentation game explained in [91]. Since the main asset to protect is customer data contained in ***fileX***, the scenario is focused towards an attempt to invalidate the confidentiality of ***fileX***. The attack scenario spans across organisational and virtual domains and is described as follows:

Scenario 1: VM image attack

AIC, a fierce competitor of ITL, has just learned that ITL has migrated to a public cloud provided by a renowned cloud provider. It is assumed that AIC already possesses some crucial information regarding ITL's IT infrastructure, most important of which is that ***fileX*** is located in ITL's VM1. AIC masterfully sculpts a plan to be able to have access to ***fileX***, which starts with creating a malware-containing image. When run, the malware is capable of producing a backdoor to the virtual machine it is run on. The malicious image is then put on sale on an e-commerce platform (e.g. AWS Marketplace). To make it look more convincing, AIC hires people and uses automated bots to fake the amount of downloads of the image and write counterfeit reviews. AIC also launches a spoof campaign on behalf of the cloud provider encouraging tenants to adopt the new image. This is done by creating a fictitious newsletter promoting the image, targeted to Mr. Big, among other customers. Since Mr. Big has the final say regarding purchases, the plan is to lure him into deciding to place this image in ITL's virtual machines in an effort to reduce expenses. To increase the chance of success, AIC could also launch a resource exhaustion attack (assuming that its infrastructure resides within the same physical server that houses ITL's infrastructure) to make sure that the image is placed in either VM1 or VM2 (the two are connected via VSW1). Once the image sits in either VM1 or VM2, the malware can be run to create a backdoor for AIC. AIC can then have unlimited access to ***fileX***.

This point forward describes the steps in Use Case 1 and 2, with occasional references to Scenario 1. The attack tree for this scenario is given in Appendix G.5.1.

The user opens the interface, which shows an empty map of the organisation. The user inserts several standard elements onto the map and draws the relevant connections. If an Archimate model already exists, this may be used as a basis instead.

During this step, traceability can be achieved by uploading the results of the manual modelling process and tagging model components that are present there. An example can be given by uploading the ArgueSecure argument sheet which contains the claim for Scenario 1 as follows.

ARGUMENTS												
ID	CLAIM		Inference Rules		Assumptions		Facts		Rebuts		Status	Flags
#	#	txt	#	txt	#	txt	#	txt		ID(s)	IN/OUT	ansferred / Mitigatio
9	C8	Create malware-containing image; Hire people or use bots to fake downloads and reviews of new image; Get target to use new image (e.g. by sending out newsletters); Wait until target launches "patched" VM into production or give it access to production data; Run malware	R9	-	A9	Amazon marketplace waive any liability w.r.t uploaded images	F9	-			IN	ACCEPTED

Figure 7.3: ArgueSecure argument for Scenario 1

Source: The TRE$_S$PASS project repository

The example shows that it is likely that not all model components can be traced back to the manual model process. According to the integration diagram, ArgueSecure arguments contribute to formulation of stakeholder goals, which are later used to construct attacker goals and formulate scenarios. However, we see this as another potential to extend the use of the trace model. Although the tagging feature might not seem very beneficial for

this purpose, the upload function can help users track back where attacker goals stem from. The upload function can also be used to attach the Service Level Agreement (SLA) of the cloud solution. We see this as a potential way to build traceability between organisational cloud-specific policies and its source; however this is not explicitly captured in the TRESPASS integration diagram. Nevertheless, this could be of added-value to the TRESPASS process.

The user clicks on the items to select the corresponding properties, which may consist of asset values, potential damage upon failure, access control mechanisms, and an estimation of their strength. Default values can be used if information is unavailable.

The TRESPASS model for Scenario 1 is shown in Figure 7.4 as follows. In the model we see the `Customer_Admin` for ITL added as a new actor. Since a cloud adminstration has access to the customer's entire virtual infrastructure, an organisation should always consider the cloud adminstrator as an actor for the analysis. Whether the administrator is regarded as a threat agent is left to the modeller. There are indeed published works that assume cloud adminstrators as a malicious actor [24] [39], being particularly dangerous because of their high skills and high access privilege to the infrastructure. Currently the ANM demo supports classifying actors into one of the following categories: *Actor Management, Actor Employee in IT, Actor Employee (trained in social engineering awareness), Actor Employee (untrained),* and *Actor Other.*

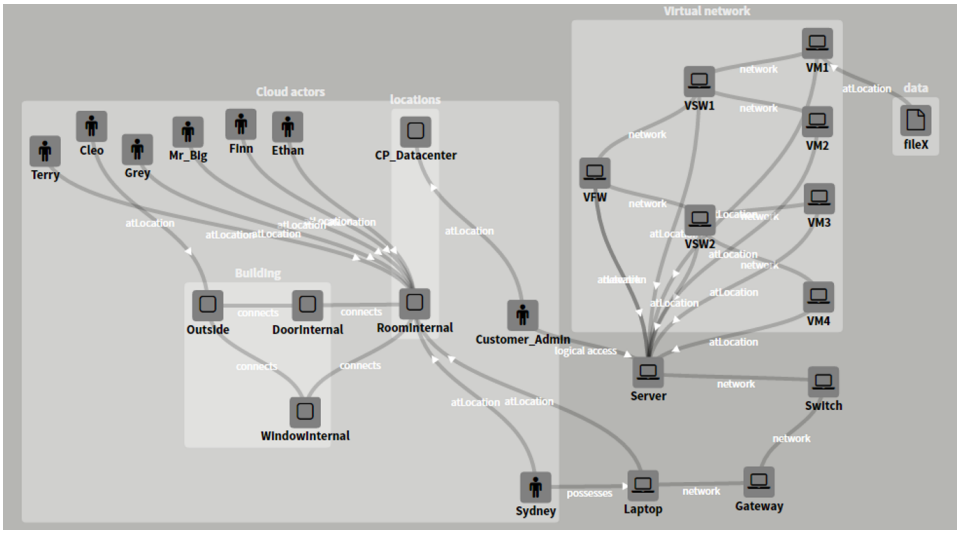


Figure 7.4: TRESPASS model for validation case, Scenario 1

We learned that from all the actor-specific parameters shown in Figure 6.5, only the parameter `susceptibility_se` is accounted for analysis (`soc_eng_probability` in the knowledge base). The decision within the project is to keep the parameter not instance-specific, but instead attached to a class where a certain value is associated with it. However, the metadata based on SP2, SP3, SP4, SP5, and SP6 can still provide more context as to why an actor is categorised under a certain class. For example, if an actor is marked as a trained employee in social engineering, `data_type` and `data_source` could indicate when and where the training took place. With regards to assets such as laptop, only the operating system is being taken into account. The metadata can still help understand its properties (e.g. `data_type`: Technical data and `data_source`: System administrator archive). It is a good idea not to make every field mandatory to be filled in, as some might only apply to specific components.

Because the current demo environment does not take many asset-specific parameters into account, it is hard to demonstrate the difference between an outsourced cloud context and a private one. However, should the project decide to add more specific parameters with regards to the virtual infrastructure, we see the SLA, or the contract between the customer and the provider as a valuable source of data to provide technical information. In a typical SLA, technical issues such as availability, performance, data security, disaster recovery expectation, etc. are addressed[3]. The metadata fields we have devised can be used to register these technical information and help clarify the source of specifications of the virtual infrastructure. However, the customer will still lack knowledge of the *physical* infrastructure, as there is little information on their side with respect to the storage location of their data, which is a typical problem with outsourced cloud infrastructure. This will create problem if the customer has to include specifications of the Server.

[3] `http://www.wired.com/insights/2011/12/service-level-agreements-in-the-cloud-who-cares/`

With regards to SP19 where we seek to establish a logger, a metadata in the form of timestamp is generated as part of a standard process in registering all vulnerabilities that are identified in a specific asset. An investigation into the knowledge base reveals that vulnerabilities, instance parameters, and model persistence files are equipped with logging information. They are presented in Appendix G.5.3.

Identification of library elements is currently being done differently from how we proposed in SP9. The demo assigns special IDs to attacker profiles with the format of `APxxxx`. There currently exist 11 different attacker profiles. With regards to actor archetypes, the identification mechanism works under the assumption that labels are unique, eliminating the need to assign another specific metadata for identification. The labels of categories of actor provided also act as identifiers (e.g. `Actor Employee (untrained)`). For model templates such as building patterns or cloud rooms, there also exists a unique, automatically-generated ID, for example `id-SyOlmwhKQ-group`. The suffix `-group` indicates that this template involves a group of different components.

Finally, discussion sessions with experts reveal that SP17 would not be very necessary and therefore might need to be reconsidered. Such specific instance-based values as we provided in the design example (`susceptibility_se`) currently are not supported. This value is based upon the class that an actor is assigned to. For instance, `Actor Management` is associated with a probability of 0.6 of falling prey to a social engineering attack. With higher availability of social data, more actor categories could be added with more precise values of likelihood, but it is the current decision within the project to keep these values simplistic and class-specific instead of instance-specific.

The user answers a number of general questions about the organisation in survey style (e.g. organisational culture). The answers are used to set properties of the items of the map, in particular the people/employees.

We skip this step for the testing as there are no specific criteria derived from this step.

The user inserts additional information regarding the model components in plaintext format (submenu 'Comments' or something similar). This information could be the context of the organisation when the model was created, the purpose of the analysis, important assumptions, components exclusion and inclusion decision, etc.

The discussion with experts yielded some suggestions to improve the feature of affixing comments to the TRESPASS model, for example to give headings under the general comment sub-wizard to guide the user in including important notes. So far we have devised the following sub-headings: **Context**, **Purpose**, **Assumptions**, **Abstraction Level**, and **Others**.

Under **Assumptions**, the user can input general assumptions on the physical infrastructure or attach them to the nodes, should he wish to. The **Abstraction Level** section may contain rationales leading to inclusion/exclusion of parts of the infrastructure, for example the modelling decision that only takes into account the user's infrastructure and not other tenant's because there is no visibility towards other tenants. The user may also explicitly state the types of assessments left out in the analysis, for example privacy assessments. In **Context**, the user can include information from the SLA that can help other people in the organisation to understand the modelling context, for instance the location of the virtual infrastructure and access to the data. Finally, the user may wish to explicitly mention the expected 'validity period' of the model under **Others** due to constant changes that happen to the virtual infrastructure, or mention that the assessment is focused towards possible attack scenarios from the competitor (AIC).

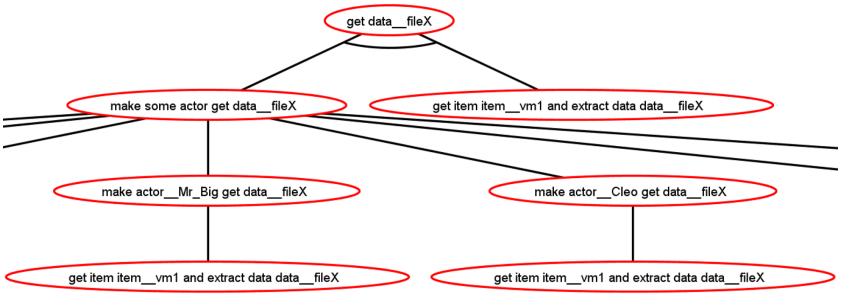
The user answers a number of general questions about the threat environment. The user can select the attacker profiles of interest, and adapt these if necessary.

It is not a far-fetched consideration to assume that the Customer Administrator is an 'insider threat', in the sense that he/she is already knowledgeable of ITL's infrastructure. As mentioned, currently the identification of actors are done on a class level, meaning that there exists no specific identification of an instance-level. The combination of the class ID and the label is the unique identifier of each actor node. This also applies with attacker profiles. A unique ID as we propose in SP18 does not exist in ANM. In this case, we could classify Customer Administrator under the class of `Outsider Trained` which has the ID of `AP0009`. When needed, the user can provide the reasons behind this selection in the node-level comment (previous step).

Aside from the customer administrator, the usual threat environment that ITL faces still applies: competitors, cyber criminals, etc. Our takeaway here is that in an outsourced cloud context, the threat environment gets potentially wider with the addition of the cloud administrator, who may or may not be perceived as a dangerous entity.

The user starts the analysis and selects one or more of the following results (1) A list of possible attack scenarios, ordered by their risk; (2) A list of possible countermeasures, ordered by their cost-effectiveness; or (3) A visualisation of the map, showing 'the weakest links', for example by increasing their size, or changing their colour.

Based on verification with experts, the unique identification for the parameters in the APLTool would now include the ID of the node in which they are contained. This is shown in Appendix G.6. These node IDs would also be shown in the visualisation results (SP24), so that users can understand the results in terms of the attack tree. From verification with experts we also realise now that Tree Maker has already enabled the embedding between model components and attack tree as we propose in SP20. A mock attack tree generated from the ANM demo[4] shows this embedding as follows. This labelling pattern is retained until the attack tree gets annotated by the APLTool.



As the augmented and annotated attack tree enters the analysis stage, ATA analyses the attack tree in order to look for the optimal attack vector combination. During the verification sessions, it was revealed that the input format of ATA has been changed so that the parameters associated to different values are crystal clear (cf. Appendix G.5.2). Finally, the storage of the parameters resulting from augmentation and annotation of the attack tree is done during the call of the toolchain. These parameters are checked into the Git system for later usage by the ANM.

In sum, processes involved in this step remain the same both in a private cloud context and an outsourced cloud context.

The user selects the scenarios/attack trees to understand which components are linked to a specific scenario/attack tree. Upon clicking on a certain leaf node, the model shows model components that are involved, such as assets (annotated with their quantitative values), control mechanisms, and actors.

We have learned from the previous step that embedding of components to attack tree labels is supported in the current TREsPASS process. However, we also learned that no visualisations of attack tree are brought to the user during the process until the visualisation stage arrives. Therefore, SP16 is dropped from the list of specifications as it is not aligned with the TREsPASS workflow. Finally, the user can also search components using the search bar functionality (SP25), which would most likely be placed in the visualisation results screen. This step is in essence a continuation of the previous step.

The link from the previous step brings us to the conclusion that processes involved in this step do not display differences between a private cloud context and an outsourced cloud context.

The user annotates explanations of security decisions next to the analysis results in plaintext format. For example, the user can type in one or more security decisions that will be made in the next three months (or any time period) which corresponds to preventing a specific attack step/scenario. This explicitly shows couplings of decisions and the danger to be mitigated. Alternatively, the user can also provide a general list of possible countermeasures, among which the user selects a few to be applied within the next six months (or any time period).

In a common outsourced cloud situation, there is a fair share of risk sharing between the customer and the provider. Although oftentimes such an agreement remains vague, the SLA facilitates discussion on important points of liabilities regarding security risks for each party, for example provider's guarantee of redundancy to help sustain connectivity (closely related to downtime and availability).

We have specified the following items that can potentially be noted down in the commenting feature: *Next steps to be taken* listing action items related to the decisions taken, *Project champion* to be held in charge for the

[4] `https://trespass-tkb.itrust.lu/anm/`

deployment of the program and contact persons, and *Next cycle* to control the decision. Following up on the suggestions of the experts, a devoted section for discussing *vulnerabilities* would also be helpful, although this is not unique to the outsourced cloud context. Adding an *Others* section is always helpful to cover points that do not fall into one of the aforementioned categories. Considering the variety of issues that might have to be dealt with, each topic can become independent headings within the comment section. Readers should be aware that security decisions entail more than investing in countermeasures, but also for instance decisions to *accept* certain risks (that are deemed as negligible according to the analysis results).

The lesson learned in this step is that in an outsourced cloud context, many security decisions will involve the cloud provider and therefore this feature becomes increasingly important (1) to clarify the shared security responsibilities between the organisation and the cloud provider and (2) to establish accountability that these security decisions are taken within the context of the results and that when things go haywire, there is written proof that such decisions were mutually agreed. We also believe that within such context, the general comments would be more interesting than the node-level (or attack path-specific) comments. Possibilities for importing the comments (and the visualisation results) could be considered so that the cloud provider would also have visibility to the results (of course by taking into account data confidentiality concerns).

The model can be saved for future references or adjustments.

7.3.2. Scenario 2 exercised in Use Case 3

As is the case with Chapter 6, Scenario 2 is meant to help use explore the specifications related to the dynamic view goal. In this Scenario, we select one type of possible change from the various types of changes that could happen in the organisation as acknowledged in Section 5.2.3 and apply it to Use Case 3.

Scenario 2: Changes

For this case study, we envision one type of change, which is *changes in attacker profiles*. Based on a recent attack attempt in HGZ, Inc. - another cloud-utilising organisation - ITL would like to rerun an analysis while taking into consideration an alleged type of attacker profile that was not considered previously, which is **Vendor**. In the attack attempt on HGZ, a third-party hardware contractor had tried to penetrate into the virtual infrastructure early in the morning to steal a highly confidential file. The attempt was discovered after the customer administrator discovered an unusual login attempt outside office hour. The drive behind the attack is still under investigation, but it is highly suspected that a competitor of HGZ has paid a generous bribe to the contractor. In accordance with this, ITL has decided to include every third-party contractor it is working with to have a more robust risk assessment. Of course the attack attempt itself can be regarded as a change that should be considered, but since ITL has very little information regarding the attack steps, it decided not to regard this as a change for the time being.

The user receives push notifications from one of the available data extraction tools or requests automated data extraction. The push notifications inform the user that there have been updates in the data based on changes detected by the input tools (see Step 3)

Firstly, the new attacker profile **Vendor** is inserted into the knowledge base in XML format with the following declaration:

```
<profile id="AP0012"
codename="Vendor"
description="Business␣partner␣who␣seeks␣inside␣information␣for␣financial␣advantage
over␣competitors"
budget="40000"
skill="M"
time="D"
/>
```

A pop-up notification indicating that a new attacker profile has been inserted will show up. This push notification is however not unique to the outsourced cloud context. In a private cloud context this type of change is also possible.

The user opens the file containing the model used in the previous assessment. The file also contains the previous analysis results.

Following up on the push notifications, the user adapts the model where needed to represent the changes observed, by moving, deleting, or adding elements and changing properties.
In response to the new attacker profile added, ITL adds the administrator of its current printer service into the model, named Printer Admin. The printers in ITL work with intranet and are maintained by a third-party service provider. Although the service administrator has limited access, ITL is worried that he could bypass the virtual firewall and gain access to the virtual database server stealthily. For the analysis, the Printer Admin is assigned the attacker profile **Vendor**. The user can annotate comments as to why he is assigned under that attacker profile and not other possible profiles such as Outsider Trained, and that he is assumed as malicious for the purpose of the analysis.

Alerts will be issued to the user when important updates have been made, or when user action is required to assist in the updates.
The Git system has now captured the addition of the new Printer Admin in the TREsPASS model. The diff between the old model and the new model can be shown in the front-end in a pop-up summary as shown in Chapter 6.

The user re-runs the analysis to identify any differences in the analysis results given the new conditions.
The Git system used in the knowledge base can register the timestamp of commit of the analysis file and version is automatically registered. The only pitfall we see from SP30 is that the owner might not be captured by the Git system. In hindsight this should not be an issue if there is only one person who does the risk assessment in the whole organisation. Otherwise, attribution problem could emerge as to who did what changes and under what reasons.

Alerts are issued when changes in the underlying data lead to significant changes in risk.
With the help of the *git-diff* functionality happening in the back-end, it is possible to capture the difference between analysis result instances saved in the knowledge base, as specified in SP31. However, a user interface is needed so that this diff result is intelligible by the user.

The model can be saved for future references or adjustments.

7.4. Analysis of single-case mechanism experiment

We start off by describing the process behind the selection of the case for the experiment. Two candidates were considered for the selection. The first candidate is a case on private cloud infrastructure with very similar physical and organisational settings to the design case. The uniqueness of this case lies in the different attack vectors described, which are more focused towards technical. Unlike the design case, this case is part of a restricted deliverable within the project, which would require special permission for publishing. Another concern that was raised was that there is a high possibility that this case would bear much similarity with the design case, therefore providing little room to reflect on. The second case which we decided to go with was seen to be unique enough for the reasons we already explicated.

Another important point we wish to convey is that the validation process is very dynamic because a large part of it depends on the ongoing progress made within the project. The verification sessions conducted with experts provided many valuable inputs, for instance with regards to the visibility to the project's decisions that have impact

on our design, not to mention the important concerns with regards to how the specifications can be refined. We also gained access to a demo environment of the ANM that has helped us understand the interaction between the user and the system, as well as the configuration of the knowledge base.

In the following sections we describe how we draw descriptive inference (from *Descriptions*), abductive inference (from *Explanations*), and analogic inference (from *Generalisations*). Finally, in *Answers* we tackle the knowledge question posed in the experiment: *Does the trace model work in different contexts of cloud computing environments, namely both in private and outsourced contexts?*

7.4.1. Descriptions

In this section we explicate how we draw descriptive inference from the case. We begin by explaining the data with which we devised the experiment. As explained, the case for experiment was retrieved from a publication on argumentation-based risk assessment activity, polished with details that bring coherence to the situation we seek (outsourced cloud). The first scenario was also derived from one of the arguments resulting from the same work, and the Use Case is essentially a repetition of the one used in Chapter 6. We safely claim that at this point no extensive data manipulation was involved, rather sticking to the 'default value'.

Apart from these publicly available data, we are also grateful for being given access to the internal TREsPASS Subversion (SVN) repository, which caters a plethora of data and work in progress that we could utilise during the validation step. The ANM demo environment also allows us to fiddle around and have a flavour of the final version of the TREsPASS tools. Last but not least, constant communication with researchers with the TREsPASS project supplied a fruitful exchange of ideas and discussions.

The profound difference from the design case throughout deployment of the Use Cases was imminent during the steps that encompass the model creation stage in TREsPASS . A lot of underlying assumptions had to be formulated to compensate for the lack of information in the customer's end. However, we could not fully demonstrate the differences of an outsourced cloud context to the a private one as the demo environment is still unfolding and many features we expect to see have not yet been placed. We also realise the importance of the post-analysis activities (making security decisions), in which we try to enlist the types of possible decisions that could be made. In our case of an IaaS setting, usually the customer is responsible for the identity and access management, its own data, its applications, and its operating system, while the provider is responsible for the virtualisation, network, infrastructure, and physical infrastructure[5]. However, we foresee some grey areas in this. For example, it could be the case that the customer discovers a certain network vulnerabilities before the provider does; how does the customer then deal with this? The security decisions should clearly address these issues. For the customer, training the administrator to skilfully handle compliance issues is almost certainly a must.

7.4.2. Explanations

In this section, we unravel the logic behind abductive inference from the experiment. Differences in the experiment are foreseen to mainly come from the architectural differences between the design case and the case for experiment. We have tried to minimise distortions coming from the treatment instrument: we employed the same Use Cases to apply the scenarios. More specifically, the architectural inference we draw is highly influenced by the abstraction level of the case. We have deliberately chosen to zoom in the customer's infrastructure and not focus on other tenants' infrastructure. Indeed, these are elements that could help provide a fuller picture of the real-world situations. For example, if the organisation is knowledgeable of how many tenants are residing in the cloud solution it inhabits, then logically the organisation would be able to predict more scenarios that could lead to undesirable situations. Another piece of information that could also help is the contract or the SLA. We also decided not to insert a mock SLA into the picture (which would be a valuable source to derive policies) because it would make our analysis busy and unfocused, which is not our intention.

With regards to changes, it may be tempting to think that the change scenario that has been devised should represent the outsourced cloud context. Here we argue that the organisation suffers from high information asymmetry. For instance, lack of visibility prevents the organisation from predicting a change in the number of tenants co-residing in the cloud solution. The organisation is confined to model only internal changes or changes that are publicly known. Through the exercise with the outsourced cloud context we find it very hard to capture the multi-tenancy nature of the cloud context and its associated risks, unless the. We argue that a complete and accurate risk analysis in such context can only be performed by the cloud provider itself, who has full knowledge of how the cloud infrastructure is configured. Unfortunately, service level agreements (SLA) often indicate that such risks are transferred to the customer, which obviously do not make complete sense for the reasons stated above.

[5] https://blog.cloudsecurityalliance.org/2016/06/14/securing-hybrid-cloud-skills-need/

An obvious limitation of an analysis for the outsourced cloud context is that it requires *every* node (actor, virtual infrastructure, location) to be modelled in order for the calculation of the possible attack scenarios. Again this is very difficult to do in such a context. From the modeller/risk assessor perspective, there are a lot of unknown information regarding the infrastructure due to multiple tenants and mixed configuration of different physical and virtual machines. A cloud customer such as ITL has neither control nor information on the physical infrastructure of the cloud provider and resource allocation which might give rise to unpredictable attack vectors (e.g. side-channel attacks). A simple example is the decision to assign to which actor class the Customer Administrator belongs. Furthermore, could he be regarded as a potential attacker? Our trace model tries to provide clarity by for example giving the space to give context on a model component. In this scenario, the model only takes into account the virtual infrastructure of ITL, simply because there is no knowledge about the virtual infrastructure of other tenants.

7.4.3. Generalisations

In this section, we finally illuminate the process behind the last type of inference we conduct: analogic inference. The intended scope of our generalisation is cloud infrastructures, regardless of their context (private/public/hybrid). We attempted to capture this intended generalisation by choosing a case that sits at the other extremity from the design case (fully private; provider and customer is the same entity - fully outsourced by third-party provider). Between the extremities lie several possible configurations in the spectrum, for example the cloud provider offers private virtualised server to the customer. By doing this, we attempt to seize elements that would also be relevant in the configurations between the extremities. However, we cannot strongly claim that this is the case; a hybrid infrastructure might have unique situations that do not appear in both a private and outsourced context.

We start by studying the similarities of the case with real-world context. We believe that the case is a very common situation in real life, where an organisation chooses to pay for an instance in a multi-tenant model. The size of ITL as an organisation makes it very possible that in real-life it would go for this type of solution, because it will benefit from the economies of scale of a multi-tenancy situation. As we have addressed in the explanation for abductive inference, we chose to keep the same Use Cases to minimise distortion coming from different treatments and we did not leave out any steps of the Use Cases as presented in the design phase. The only alteration we intend to bring as a treatment is the choice of scenarios applied to the Use Case. By employing different scenarios, we explored other pieces of the cloud jigsaw that could provide new perspectives in validating the trace model.

However, we may have left out several relevant facts in order to achieve the desired generalisation. The most important of this could be provider-specific facts. Each renowned cloud provider (Microsoft Azure, Amazon Web Services, etc.) has been placed under scrutiny by researchers, giving rise to the discovery of many specific vulnerabilities and risks that could not be found in other solutions offered by other providers. Each provider also has its own unique configuration and resource allocation model, which may sometimes be left unknown unless we decide to hire an instance of their solution. To keep our validation experiment slim, we have decided not to include these provider-specific facts in the risk assessment, but instead opted for an "imaginary" provider whose specific risks and vulnerabilities are up for exploration.

Finally, we wish to address the possibility of generalising the functionalities of the trace model to other case studies in TRESPASS . Generalisation to other TRESPASS case studies are harder to infer, since the case studies can be very context-specific. Some features and processes may apply uniquely to a specific case. Within TRESPASS these case studies are are also strongly distinguished from one another, leaving little space for them to converge. We made a conscious choice to concentrate solely on the cloud case study in designing the trace model. However, we have attempted to mix examples with the IPTV case study where applicable to show that the design criteria may sometimes generally apply to the other case studies and are not case-bound.

7.4.4. Answers

In this section we give answer to the question *Does the trace model work in different contexts of cloud computing environments, namely both in private and outsourced contexts?*. We also briefly discuss what the result of the experiment implies for the knowledge goal (*validating the trace model*) and the improvement goal. The short answer to our knowledge question is Yes. We have seen that the design of the trace model as it stands now does support the extremities of the different cloud context spectrum. The main lesson we learned is that in different situations, some features may appear to be more prominent than other features, based on the uniqueness of the situation. In order for the trace model to capture this, its feature should be flexible enough to allow new dimensions to be captured, but also have enough structure so that the user is intuitively guided towards fulfilling traceability links that we seek. Results of the experiment show that where the features are relevant to the outsourced context (e.g. features to add comments to the model and enlist security decisions), our design is flexible enough to capture the differences and might only need minor tweaks (e.g. additional headings) to satisfy its requirements. Lastly, we also

believe that the design of the trace model is closely aligned to the definition of traceability in cyber risk assessment as we have formulated in Chapter 4. This is visible in the processes and features that are strictly cascaded down from the static view goals and the dynamic view goal of traceability articulated from the definition.

The experiment combined with expert opinion is more powerful than it would have been had we not consulted experts for their opinion. Using information from the experts, we could see which features are not aligned with the TRESPASS process, and we also learned what features have been partially supported in the back-end (which implies that front-end support is where we should focus on). Other validation methods such as technical action research (which involves a real-life case study) and statistical difference-making experiments (which uses samples of different validation models) could also be applied to test the trace model.

7.5. Validated specifications

Based on the discussion with experts and the single-case mechanism experiment, we refine the specifications originally featured in Table 6.5. In Table 7.1 we indicate the original specification and the modifications done as necessary. Added details or deleted specifications are marked as (+) or (-). No preceding marks means that the modification is still up for further investigation. It turned out that the single case mechanism experiment provided less contributions in validating the trace model. We suspect that this is caused by the limitation of a tangible prototype with which we could experiment.

Table 7.1: 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)	-
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

Table 7.1 Summary of validated specifications (continued)

Code	Original specification	Modification	Source of validation
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
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

To wrap up this chapter, we reflect on how these specifications on traceability have been taken into account in TRESPASS . We begin by evaluating the role of the knowledge base as a central data repository in the TRESPASS process. The fact that every process and data (including quantitative parameters for the attack tree) virtually has a linkage with the knowledge base benefits us in enshrining the desired trace links. We have also noticed that model components are accounted for in the attack tree labels, making it possible for the user to understand the attack steps in terms of the model components and trace them back to the model. This structure gets retained until the attack tree gets annotated with quantitative parameters, enabling the users to make sense of these parameters in the context of the models. The fact that attackers are considered as part of the model (an actor can be assigned to a certain attacker profile) is worth our attention. With regards to actors, the current process is already equipped with the capability to spawn a unique identifier for every attacker involved in the analysis with the goal of being able to connect the model with the attackers used. Installing necessary metadata for managing data could undergo immediate implementation as this does not require relatively high workload. Specifications requiring further user interface development could be placed lower in the priority order, followed by specifications supporting dynamic analysis in TRESPASS . In sum, opportunities to imbue traceability in TRESPASS are still open for exploration.

8

Conclusions and discussion

This chapter sums up the findings and results of the thesis in its entirety. Answers to every sub-research question and our main research question are delineated here. Also brought into the discussion are the recommendations for TRESPASS based on the validated trace model. We reflect on limitations encountered during writing the thesis and designing the trace model. Finally, we outline some possible future research opportunities based on our findings.

8.1. Conclusion

We begin by answering the sub-research questions and research question posed in Chapter 1, section 1.3. For easy reference, we reiterate these questions as follows, starting from SRQ1 to the main RQ.

SRQ1: **How can the notion of traceability be defined and specified in the domain of cyber risk assessment?**

The answer to this question is encapsulated in the formal definition of traceability in cyber risk assessment we have formulated at the end of Chapter 4, which is:

> *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.*

Our definition implicitly states that there are two sides of the traceability coin: a *non-dynamic* side and a *dynamic* side. The non-dynamic side (in Chapter 5 onwards simply referred to as *static*) construes that traceability should be represented by bidirectional trace links spanning from data sources to the product of the risk assessment activity - risk treatment decisions. This means that both *tracing* of results and *tracking* of data should be supported, with the goal of providing rationales why certain risks apply. The dynamic side introduces the concept of changes in risk assessment and how the two can be linked together.

To the best of our knowledge, such a formal definition has not been yet formulated. The *Discussion* section extends this elaboration by comparing it to an existing notion of traceability in cyber risk assessment. Having answered the first SRQ, we move on to the second SRQ, which is:

SRQ2: **What are the requirements to provide a traceability support for TRESPASS in the form of a trace model applied in a cloud computing environment?**

We readjust our focus by taking the answer to SRQ1 and translating it to TRESPASS context. In Chapter 5 we have stipulated goals for the trace model based on the static-dynamic duality of our definition as part of the problem investigation phase in the design science approach. In Chapter 6, these goals are unfolded into requirements that the trace model needs to fulfil along with requirements tapped from interviews and Use Cases. The combined activities yielded a total of 31 requirements covering functional requirements, user requirements, and non-functional requirements. These requirements are summarised in Table 6.2. We also posited contribution arguments for these requirements. The requirements we have specified are unpacked into design criteria that enlist the processes, tools, and concepts needed to fulfil them. Using Use Cases from TRESPASS, these design criteria are diced to become design specifications, summed up in Table 6.5. This concludes the treatment design phase in the design science approach. The final phase - treatment validation- is addressed in SRQ3 as follows:

SRQ3: **How can the trace model be verified and tested to see whether it fulfils its requirements and is applicable to similar cloud environments?**

Still within the TRESPASS context, the specifications of the trace model are verified in this step to see whether they actually resonate with the requirements by generating expert opinions. We also took into account whether some of these requirements have been taken into account. They are then tested to infer applicability to different cloud computing contexts. The verification and validation activity produced information regarding feasibility of implementation in the project, how (parts of) the specifications are currently being approached in TRESPASS , as well opportunities for further improvements. This resulted in a list of validated specifications presented in Table 7.1 according to the results of the treatment validation activities.

The answers to the sub-research questions conflate to provide the answer to our main research question:

RQ: **How can traceability be incorporated in cyber risk assessment?**

The short answer to this question is by referring to the trace model that we have designed. Figure 8.1 shows an attempt to highlight the features of the trace model in each stage of the integrated TRESPASS process. The *Post-analysis stage* is distinctively marked as it is not part of the official TRESPASS process. Were we to liken the TRESPASS steps to the activities of a cyber risk assessment process, we could say that the Data Collection stage would be comparable to the Context Establishment stage. Risk Identification is covered in Model Creation stage and parts of the Analysis stage. Risk Analysis falls naturally within the Analysis stage, and Risk Evaluation activity is made possible during the Visualisation stage. Risk Treatment is eventually represented by the Post-analysis stage in our diagram. We factor in the dimension of time in our depiction of the features to show that the trace model does not only help a cross-sectional risk assessment activity, but it should assist maintenance of the activity through the course of time.

To conclude, we believe that the answers to the (sub-)research questions have fulfilled the objectives we set out in Chapter 1: *contribute to the cyber risk assessment body of knowledge by formulating a formal definition of traceability in that domain* and *contribute towards the development of TRESPASS model by generating support for traceability in the form of a trace model.* These contributions are:

- A formal definition of traceability in cyber risk assessment which exhibits both a static view and a dynamic view;
- 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;
- Demonstration of building linkage between Context Establishment stage and Risk Identification stage via identifying the entities involved in the manual risk model; and
- 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.

These contributions are explicated further in Section 8.2.2.

8.2. Discussion

Nearly every topic can fit into the discussion section, but we will restrict the discussion to subjects that we think would be interesting. Firstly, we evaluate and reflect on the expected contributions of this thesis to TRESPASS process. We also reflect on how our approach stands up against what is offered by CORAS and how the two can potentially complement each other. Secondly, we also take this opportunity to ponder upon the repercussions of our trace model for risk assessment practice in general.

8.2.1. TRESPASS-specific contribution and comparison to CORAS

From the initial semi-structured interviews, we learned that some experts suggested tool support as the ideal form for the trace model. However, we also find that language extension is necessary. The trace model would have to be versatile enough to be extended in different formats, and sometimes internal embeddings are necessary in order to connect model components between the stages. Judging from the amount of human interaction with the trace model, we can conclude that the traceability support is semi-automated. Some of the features can be pre-programmed into the tools, but some features require substantial amount of human input from the interface. This research is aimed to contribute to one of the final deliverables in TRESPASS on best practices for model maintenance.

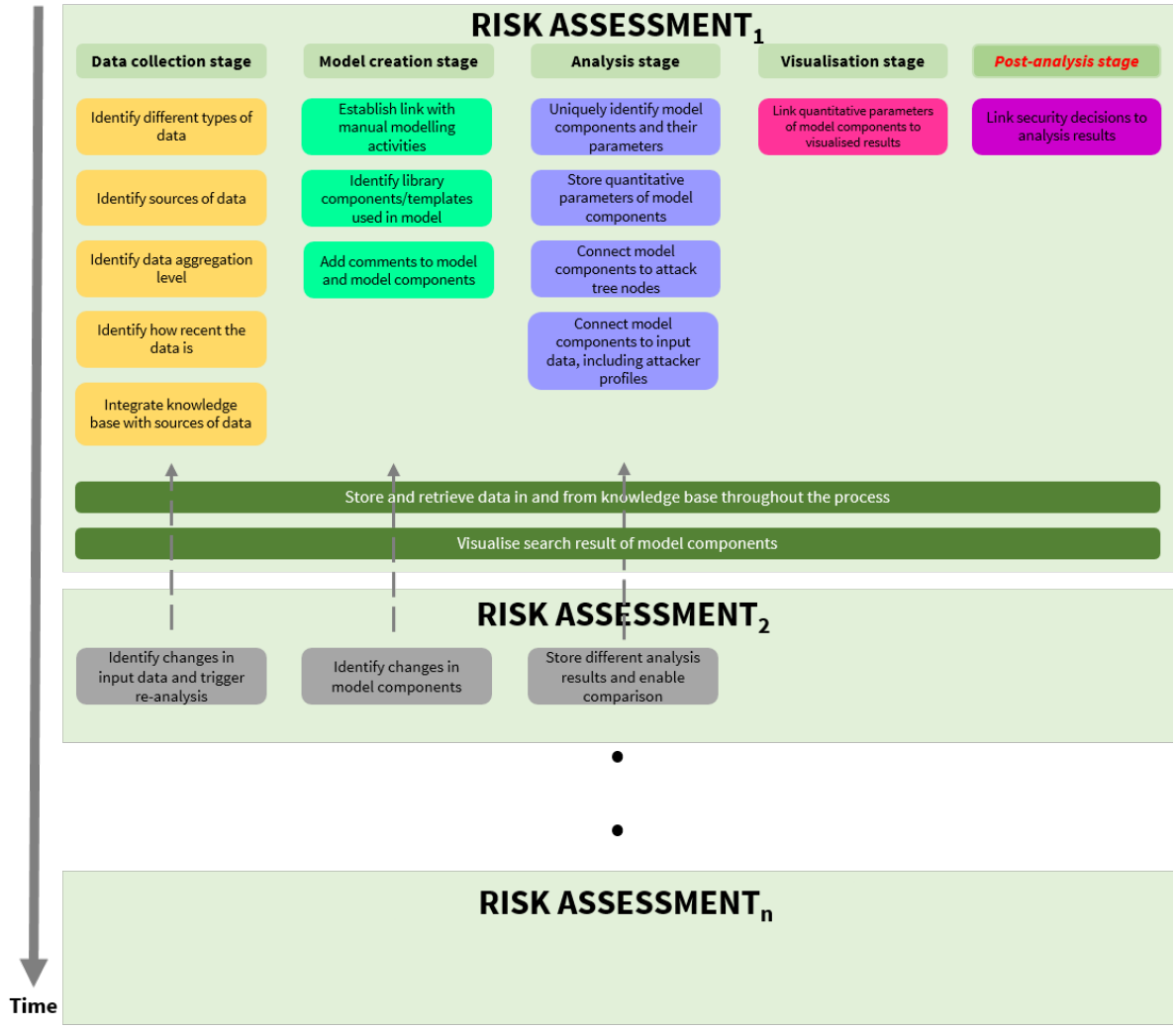


Figure 8.1: Summary of trace model features in TRESPASS

We would argue that traceability can potentially extend beyond only model maintenance. Traceability, as we have learned while composing this thesis, belongs as early as the stage of *context establishment*. It not only identifies which data would be necessary and how to maintain good record of them, but also supports a good chain of custody towards the context of the analysis.

How CORAS deals with traceability has been explained in Section 6.2. We think it is important to reiterate that TRESPASS and CORAS depart from different points with respect to traceability. CORAS' main point of departure is *change management* which includes different perspectives of changes. This change management strategy places emphasis on a mapping model that marries a risk model to a target model [147]. Our proposed traceability support departs from a pot pourri of concepts processed from literature from domains such as requirements engineering, food industry, and data provenance. Having said that, we attempt to provide a fair comparison of both these approaches.

We begin by revisiting the three perspectives of changes put forth by developers of CORAS, *maintenance perspective*, *before-after perspective*, and *continuous evolution perspective*. In Chapter 2 we have made our case of why we think that with respect to changes, TRESPASS should focus on the maintenance perspective. Our design of the trace model and its ability to handle changes is centred on this view. For example, we find it important for risk assessors to document an analysis results and be able to conveniently return to it, hence the specifications to provide metadata of analysis results (provide context of analysis) and compare different analysis results (to understand the effect of changes). To help the reader analyse how our trace model differs from that of CORAS, Table 8.1 compares the quintessential features from both approaches. A discussion thereof follows.

Table 8.1: Comparison of TRE$_S$PASS and CORAS trace models

Aspect	TRE$_S$PASS	CORAS
Format	Various (metadata, processes, language embeddings, extension to tools)	Tool support
Relation to formal definition	Captures both the static and dynamic view of traceability	Emphasises the dynamic view (changes)
Coverage	Traceability from data collection activities until post-analysis activities	Traceability between risk model and target model
Support for handling changes	Partial, not solely focused on handling changes	Focuses on automated identification and propagation of changes
Approach to handling changes	Maintenance perspective	Before-after perspective

The target model in CORAS is represented by the TRE$_S$PASS model, whereas the risk model in CORAS is analogous to the attack tree in TRE$_S$PASS (the risk model reports findings of threats, vulnerabilities, threat scenarios and unwanted incidents involving the assets in the target model). TRE$_S$PASS does not treat the trace model as a literal/separate model that bridges the risk model and the target model. Instead, the trace model framework is designed to be incorporated in the process, both in the construction of the models and the linkage of the two. Both trace models are alike in some aspects. They both intend to assist users in speedily identifying changes in the target of analysis by making use of diff features and support representation of change-impacted risks in the target systems. In the next section, we discuss the contribution of the trace model beyond TRE$_S$PASS .

8.2.2. Contributions to risk assessment body of knowledge

This section first discusses the static view and dynamic view of traceability as we discovered in our research, then we move on to a discourse on the benefits of traceability to risk assessment practice in general.

As we have acknowledged early in this thesis, fixing the lack of understanding of what traceability in risk assessment entails is our point of departure in this research. Our findings suggest that traceability does not go only as far as changes are concerned; it implicitly demands to be present even *before* the actual risk assessment process begins and *after* the risk assessment process is finished. This is because traceability covers a lot of grounds in the risk assessment "lifecycle": starting from managing and keeping track of the necessary data until assisting (security) decision makers to relate the results to their decisions. A new cycle could resurge when these decisions are used as inputs to perform another risk assessment and it basically continues down the road. As we show in Figure 8.1, it is important to acknowledge the importance of traceability in an instance of a risk assessment activity, but equally important is to see how it fits in the bigger picture of dynamic risk assessment. In Chapter 1 we have stated that risk assessment is only a part of a continuous risk management process; *communication and consultation* and *monitoring and review* activities are also important to ensure the continuity of an organisation's risk management process. Refsdal et al repeatedly stress the importance of a good information system to support these two activities [147]. The aforementioned traceability feature for information sharing can for example help the organisation refer to the appropriate databases and repositories of emerging risks, if such databases do not exist internally within the organisation. The feature to keep risk treatment strategies (security decisions) can help organisation establish a proper periodical review mechanism to evaluate the effectiveness of applied controls and inform all relevant stakeholders of the risk assessment status. The feature to inform users of changes in external data sources, can act as trigger to re-monitor and review the risk assessment process. Eventually, the ability to compare analysis results in terms of countermeasures or most possible attack vectors will help stakeholders conduct reviews and draw conclusions on the progress of the organisation's risk management process in a speedy manner.

The features of the trace model can be generalised for usage beyond TRE$_S$PASS . Using the list of scientific contributions given in Section 8.1, we frame the quintessential features in generic risk assessment concepts. We start with the application of metadata in various stages of risk assessment process. Within Context Establishment, the use of metadata is especially helpful in classifying information and identifying its source. During the interviews, we have repeatedly heard that expert judgement is an important input in establishing the context of risk assessment. This is also the case in CORAS [57]. The usage of metadata can help risk assessors identify the experts that contribute to the risk assessment and the context in which the information was given. Our application of metadata in supporting

risk assessment can complement existing research on metadata-based risk assessment[1] and provenance-based risk assessment, which heavily relies on Dublin Core Metadata Initiative (DCMI) [32]. In the latter case, our proposed metadata fields provide deeper insight into the data employed. Risk Identification is commonly associated with creation of a model of the target of analysis of risk assessment. Often included are the relevant assets, actors, threats, and vulnerabilities as elements in the model. Metadata can be used to identify these instances and provide unique identification to activate provenance in the risk assessment. Metadata can also support Risk Analysis activities by providing context to external data used in the analysis such as `timestamp`, `origin`, and `version`. Data from public repositories such as OWASP (The Open Web Application Security Project)[2] and OVAL[3] can be clearly identified when used during the Risk Analysis. Ultimately, metadata records such as `timestamp` and `version` help provide overarching context of the risk assessment process and answer questions such as when it was conducted, has there been previous assessments of the same target of analysis, etc.

One of the goals of the Context Establishment is to define the internal context, external context, and the goals and objectives of the risk management process. For this purpose, meetings with stakeholders related to risk management are held. Outcomes taking the form of minutes of meeting or a simplified diagram or model of the organisation will later be used as input. When using computerised risk assessment tools, these models or decisions will have to be translated into meaningful information to feed the Risk Analysis process. The trace model to support tagging and establishing linkage between manual model and 'automated' model of the target of analysis is useful in this case. Risk assessors can understand why certain entities are included in the model based on the stakeholders meetings and similarly during Risk Identification stage risk assessors can conveniently return to the manual model to make sense of the 'automated' model.

During Context Establishment and Risk Identification, text-based explanation can be used to delineate the scope of analysis and determine the abstraction level necessary to perform the risk assessment. Such explanation can also be utilised to express focus on a particular component of the organisation (asset or actor) and noteworthy assumptions that underlie the analysis. Risk Treatment decisions can also be substantiated in text-based explanation that include choices of controls and their justification based on the Risk Evaluation results (assuming that risk mitigation is the chosen treatment), timeline of monitoring and evaluation of controls effectiveness based on the Risk Treatment decisions. Virtually any Risk Treatment decisions, including risk sharing, risk acceptance, and risk avoidance can be traced back to the Risk Evaluation control provided enough explanation exists. Our choice to implement free text field to support this feature offers flexibility in accommodating this purpose.

Finally, these benefits of traceability in risk assessment certainly come at a price. We think one of the most obvious handicap is the decreasing ease of use or usability of the system (it is assumed that most cyber risk assessment activities are performed with help of a computer-based tool). By having users performing additional tasks for traceability purposes (e.g. inserting comments). To overcome this problem, these additional tasks may be positioned as optional activities, aside from what is minimally needed to perform risk assessment. In other words, system developers should find the perfect balance between traceability and usability to produce the optimum level of efficiency of the risk assessment system. Researches on user experience (UX) may be beneficial in achieving this. Reduced efficiency is another foreseeable by-product of enforcing traceability. Due to the amount of user input required, a risk assessment can potentially become a verbose process. One way to prevent this problem is to automate the tasks needed to support traceability such that scalability is made possible. These include metadata generation, automatic tagging of entities in manual model, and so on.

8.3. Limitations

There were obviously limitations encountered during the process of compiling this thesis. One of the most imminent limitation is the fact the project will end in October 2016. This decision cascaded down to our decision of not taking implementation into account. We were also locked in to prior decisions that have been made within the project. A case of this is when we learned that parameters for Actors are kept within classes and not instance-unique, which therefore revokes the need to have specific data ID for the parameter on actor's susceptibility. Another limitation was that our research is done in parallel with the development of the interface prototype and therefore we were limited to envisioning how our features would substantiate in terms of the GUI of the tools. Finally, we could only address one case of a socio-technical system in this thesis. To produce a versatile framework for traceability in risk assessment that is applicable to *every* socio-technical system is to bite more than we can chew. Even within the TRE$_S$PASS project, we could only cover one of the five case studies (IPTV, ATM, telco and e3 fraud are not considered). We hope

[1] `http://www.oclc.org/content/dam/research/events/2012/10-02e.pdf`
[2] `https://www.owasp.org/index.php/Main_Page`
[3] `https://oval.cisecurity.org/`

at some point our work can inspire those who are interested in the area of socio-technical security/risk and develop a more bespoke approach to other domains. Lastly, we created the trace model with a tool-based risk assessment model in mind. Paper-based risk assessment would require manual workarounds to establish traceability, although in principle some of our ideas can be applicable to such an exercise. Keeping information of data in separate documents (comparable to metadata) and documenting risk treatment decisions carefully help preserve traceability of the exercise.

Keeping in mind that implementing the trace model to TRESPASS during the last few months of the project would be a tough call, we would like to offer recommendations on how traceability should become an inseparable part of the risk assessment. In general, risk assessment processes demand an abundant supply of data in various shapes and sizes coming from a multitude of sources and spanning a wide time horizon. Meticulous data governance becomes obligatory, and the concept of traceability could give risk assessors or decision makers pointers on where to start. A feedback we hear during the semi-structured interviews is that knowledge sharing on risk assessment is increasingly becoming best practice in industries. Establishing traceability on data management can facilitate the knowledge sharing activities by drawing links to the relevant databases and also by setting the appropriate level of sharing permission intended for a wider audience. At the end of the risk assessment process, traceability can also serve as a cornerstone for evidence-based security, in case there is a need to revisit the results in the future. An example of this is traceability to help understand the context behind the modelling process and why certain parts were abstracted out, when the very parts that have been left out contributed to an unwanted event. For these reasons, we believe our research has a high practical relevance in high-tech organisations. In sum, we encourage risk assessors to infuse traceability into their assessments for the aforementioned benefits it can potentially bring. While what we have done covers a wide spectrum of application, we also think working incrementally with one part at a time (e.g. with data management or with supporting decision making) is also possible and to a certain degree depends on the need of the organisation.

8.4. Reflection and possible future research direction

In this section we briefly reflect upon the process leading to the completion of this thesis. As a personal reflection, we realise that the writing process of this thesis has involved a lot of creative process. There have been difficult times when we think that our ideas are mere wishful thinking. However, consulting experts has helped us in understanding how our ideas can be refined. This happened iteratively. Hence, we believe that as with other design problems, there is not a one-size-fits-all design for the trace model and that there are ideas in other forms that could fulfil the same objectives we were trying to achieve. An evident and practical continuation of this work is to put the design into implementation and evaluate it in an iterative process. Further development can be done by referring to established standards, for example addition of metadata as necessary according to DCMI. Text-based explanation on context establishment (data collection stage in TRESPASS) and risk treatment decisions can also be developed further by providing structure in the language such that parsing is enabled and meaningful bits can be juiced out of the text.

We hope that our ideas can be propelled in follow-up researches after TRESPASS is finished, for instance researches that would fit under Horizon 2020 call on Digital Security Focus Area. Upcoming topics that could be relevant include *Addressing Advanced Cyber Security Threats and Threat Actors* (DS-07-2017) and *Privacy, Data Protection, Digital Identities* (DS-08-2017). Our ideas on identifying changes in input data (e.g. new threat agents) and data management can contribute to researches on the aforesaid topics. We also hope our ideas can serve as design guidelines for projects with similar objectives, such as the projects *CISP (Collaborative and Confidential Information Sharing and Analysis for Cyber Protection)*[4] and *PROTECTIVE (Proactive Risk Management through Improved Cyber Situational Awareness)*[5] (both due to commence in late 2016). For example, CISP aims to "facilitate the definition, analysis, management, enforcement and dissolution of data sharing agreements; going from high level descriptions (close to natural language) to system enforceable data usage policies". Our ideas on establishing various metadata for managing input data (such as `Permission`) can serve as pointers to achieve this goal. One of the goals in PROTECTIVE is to rank "critical alerts based on the potential damage the attack can inflict on the threatened assets and hence to the organisations business. High impact alerts that target important hosts will have a higher priority than other alerts". Our idea to compare different analysis results can be translated into comparison of alerts and help decision makers make sense of their relative importance and assist prioritisation.

In a more abstract vein of research, researchers may start considering traceability in (model-based) risk assessment as a distinct value to pursue. In Chapter 6, we hinted at the notion of techno-regulation to achieve traceability

[4] `http://cordis.europa.eu/project/rcn/202687_en.html`
[5] `http://cordis.europa.eu/project/rcn/202674_en.html`

by design or traceability by default. Pursuant to this end are a lot of exploratory questions. One instance is to investigate how value sensitive design can incorporate traceability in risk assessment from the pre-design stage. Friedman et al define value sensitive design (VSD) as an approach to design technology that considers human values throughout the design process [63]. Privacy, universal usability, freedom from bias, autonomy and accountability are some values that are commonly implicated in the design of information systems [63]. Others add security, trust, ownership, and human welfare to the list [84]. It is yet unorthodox to consider traceability as a separate value in itself.

Therefore, can stakeholders in risk assessment view traceability as a value? Should traceability be considered as a value in itself, or a meta-value that lays foundation for other commonly known values, such as transparency and accountability [85]? How does it interact with other values? How can traceability enable good governance of (any) risk assessment process? Where does it role lie? Finally, could traceability as a value conflict with other values in risk models, such as usability or efficiency? What kinds of trade-off can be made? These pointers show that traceability has a potentially strong connection with risk assessment, albeit under-researched. We already see a growing interest in addressing the relation between values and risk assessment: the research project *CANVAS (Constructing an Alliance for Value-driven Cybersecurity)* which will also start in late 2016 attempts to imbue values and fundamental rights into technology development in cyber security[6].

Cyber risk is a real and growing threat. Over the years, in the spirit of the Fourth Industrial Revolution and the prevalence of cyber-physical systems, the role of traceability (or whatever domain-specific jargon applies) in risk assessment will become more crucial for organisations. We are certain that in the coming years the link between traceability and risk assessment will become more relevant, such that traceability potentially becomes an emerging area of interest within risk studies. We do hope that our research has provided the reader a foretaste of this.

[6] http://cordis.europa.eu/project/rcn/202697_en.html

A

Glossary of TREsPASS core concepts

The following section presents the definitions of core concepts used in the TREsPASS project, taken from D6.2.2 [195]. Definitions are up-to-date as of March 2016.

A.1. Socio-technical system

Socio-technical system

A *socio-technical system* is a system consisting of human behaviour, technology and the policies that influence human behaviour. The key properties in the socio-technical system are entities, interaction possibilities, and quantitative properties associated with interactions. The quantitative properties include difficulty, risk for attacker, rewards, visibility, excuses.

Socio-technical security model

A *socio-technical security model* is a model of a socio-technical system specifically focused on security risks in such systems, consisting of (1) a specification of objects, relations, and capabilities, (2) a specification of possible transformations, and (3) a specification of what constitutes an attack. The socio-technical security model is made up of several types of components:

Spatial components

This refers to the geometric representation of its shape in some coordinate space. This is referred to as its geometry.

Social components

A human as an entity that interacts in the model. The human can change location between rooms and can have relations with entities. Humans are actors who can be malicious or not and can interact with each other.

Locations

Entities in the spatial component.

Object component

The set of all objects.

Objects

Entities that can be moved around between locations.

Digital component

This concerns all programs and data that are present in objects supporting digital data storage, processing and communication.

Entity

An *entity* or *element* is a part of the socio-technical system that is considered separate for the purpose of analysis.

Action

An *action* is a change to the state of the socio-technical system as represented in the socio-technical security model.

Actor

An *actor* is an (in)animate object that executes actions.

Asset

An *asset* is an object related to the organisation that, when unduly accessed, modified or made unavailable would cause harm to the organisation.

Policy

A *policy* is a rule regulating access to assets.

Organisation

An *organisation* is a socio-technical system consisting of people, buildings, computers, and data, that needs its assets to achieve its goal. An organisation often formulates policies as operationalisations of the requirement of being able to achieve the organisation's goals. An attack will typically violate such policies.

Vulnerability

A *vulnerability* is a condition in a system, or in the procedures affecting the operation of the system, that makes it possible to perform an action that violates the explicit or implicit security (or survivability) policy of the system.

A.2. Risk

Threat

A *Threat* is anything that is capable of acting in a manner resulting in harm to an asset and/or the organisation.

Threat event

*Threat event*is a threat acting in a way that causes damage to an organization.

Threat event frequency

The *Threat event frequency* is the frequency with which certain Threat events occur.

Loss

Loss or *Damage* is any harm inflicted upon the organization which can take various forms such as:

Productivity: a reduction of the organisation to effectively produce goods or services in order to generate value

Response: the resources spent while acting following an adverse event

Replacement: the cost to substitute/repair an affected asset

Fines and Judgements: the cost of the overall legal procedure deriving from the adverse event

Competitive Advantage: missed opportunities due to the security incident

Reputation: opportunities or sales due to the diminishing corporate image following the event

Loss event

A *loss event* is the occurrence of loss due to the occurrence of a threat event.

Probable Loss Magnitude *Probable loss magnitude*(also called impact) is the damage that occurs when an attack scenario succeeds. A Threat Event may have impact either on one security dimension (confidentiality, integrity, availability) or on several dimensions and this impact may be complete, high, partial, low, etc. depending on the target (e.g. compromised asset, crashed service, etc.).

Risk The *risk* associated with a type of threat event is the frequency of occurrence of loss events due to this threat, times the expected impact of a loss event.

A.3. Attacks

Attacker

An *attacker* or *adversary* is an actor with goals that, when achieved, would harm the organisation. An attacker may cause threat events by executing actions aimed at achieving his goal. An attacker can be described by the following attributes:

Attacker goal is expressed as utility functions, mapping possible outcomes of attacks to (e.g. monetary) value for the attacker. An utility function is the minimal data needed to express attacker goals. The utility for the attacker upon reaching a goal may or may not be different from the (negative) utility (i.e. value of the asset) to the organisation (impact).

Attacker resources are the total amounts of time and money available for the attacker to invest in attack scenarios.

Attacker investment is the amount of resources (time and money) that an attacker decides to apply to the execution of a specific action or attack scenario.

Attacker skill refers to the attacker's skills in relation to a high likelihood of successful attack. Characteristics are often described today in terms of "script kiddie" or "geek". The skill level is assumed to be constant over the course of the attack (analysis). A more skilled attacker has a higher likelihood of success with fewer resources. Different skills are required for human based (e.g. dumpster diving, impersonation, technical support, shoulder surfing, piggybacking, etc.) than for computer based activities.

Attacker strategy describes the decisions made by the attacker based on the expected utility of scenarios. A stealthy strategy is one in which an attacker chooses an attack approach that balances the likelihood of success with the likelihood of detection: attacks that have a reasonably high likelihood of success and at the same time an acceptable likelihood of detection.

Attacker profile An attacker profile includes attacker motivation/goals, strategy, capabilities, resources, knowledge of the system and initial access. The attacker profile may include a general attacker skill level, or different skill levels for different types of attack steps (e.g. technical or social.

Attack

An *attack* is a sequence of one or more attack steps intended to achieve the goal of an attacker.

Exploit

An exploit is an action consisting of a single-step (atomic) exploitation of a single vulnerability by an attacker.

Attack step

An attack step is an action (exploit or other activity), potentially available to an attacker in order to reach his goal, and considered atomic for the purposes of analysis. An attack step has an associated:

Attack step difficulty indicates the resources and time that a particular attacker would have to spend to achieve a certain likelihood of success. Difficulty thus defines how hard it is to do the action or exploit the vulnerability.

Attack step execution is the execution of an attack step by a specific attacker or attacker type as part of a campaign toward the adversary's goal.

Likelihood of success of an attack step execution is a value between 0 and 1 indicating the expectation value of the outcome, where 1 is success (access acquired) and 0 is failure (in Open Group terms this is called "vulnerability (level)"). The likelihood of success is dependent on the difficulty of the attack step, the skill of the attacker, and the resources spent by the attacker.

Attack event

An *attack event* is an action that contributes to the adversary's goal, but is not controlled by the attacker. An attack event has an associated mean time to failure, which indicates the average time an attacker would have to wait before the event occurs.

Attack scenario

An *attack scenario* or (composite) attack is a (partially ordered) collection of one or more attack steps and zero or more attack events, leading to an attacker's goal, and constituting a threat event for the organisation. The likelihood of success of an attack scenario depends on the likelihood of success of the steps of which it is composed.

Attack execution is the execution of an attack scenario by a specific attacker or attacker type as part of a campaign toward his goal.

Threat Capability

The *threat capability* an attacker applies in an action is the combination of his skill and his investment.

Countermeasure

A *countermeasure* is a means to reduce the risk of attack, typically by decreasing the attacker's expected utility. A countermeasure can be technical (for example, the use of encryption to protect a communications link) or procedural (for example, the implementation of dual controls) or physical (for example better locks on doors).

Reward

The *reward* for the attacker consists of the expected benefit when succeeding in getting access to a certain asset.

Utility

The *utility* of an attack scenario to the attacker is the expected value he obtains from executing it. This value depends on the expected likelihood of success, the expected reward upon success, the expected investment, probability of detection, etc. How these components are mapped to a utility value is dependent on the attacker profile (see above). This can be expresses as a utility function in the attacker profile. The utility for the attacker upon reaching a goal may or may not be different from the (negative) utility (i.e. value of the asset) to the organisation (impact).

Excuse

An *excuse* relates to the social environment in which attacks occur. Depending on the subcultures within an organisation, it may or may not be clear what is acceptable behaviour and what is not. Excuses may influence attacker strategy.

Provocation

Provocations may occur if the location and means of access of valuable assets are easily visible to the attacker. For example, laptops present on the ground floor are visible through a window. Such provocations may influence attacker strategy.

A.4. Navigator Maps and Attack Trees

Attack Tree

An *attack tree* is a hierarchical, graphical diagram for representing and analysing an attack scenario.

Attack navigator

An *attack navigator* is a tool to support prediction, prioritisation and prevention of complex misuse scenarios.

Attack navigator map

An *attack navigator map* is a graphical diagram which presents the relevant infrastructure in relation to the relevant assets. All possible attack scenarios that allow attackers to achieve their goals can be generated.

Visualisation

Visualisation is either a) forming a mental image of something or imaging it, or b) making something visible to the eye.

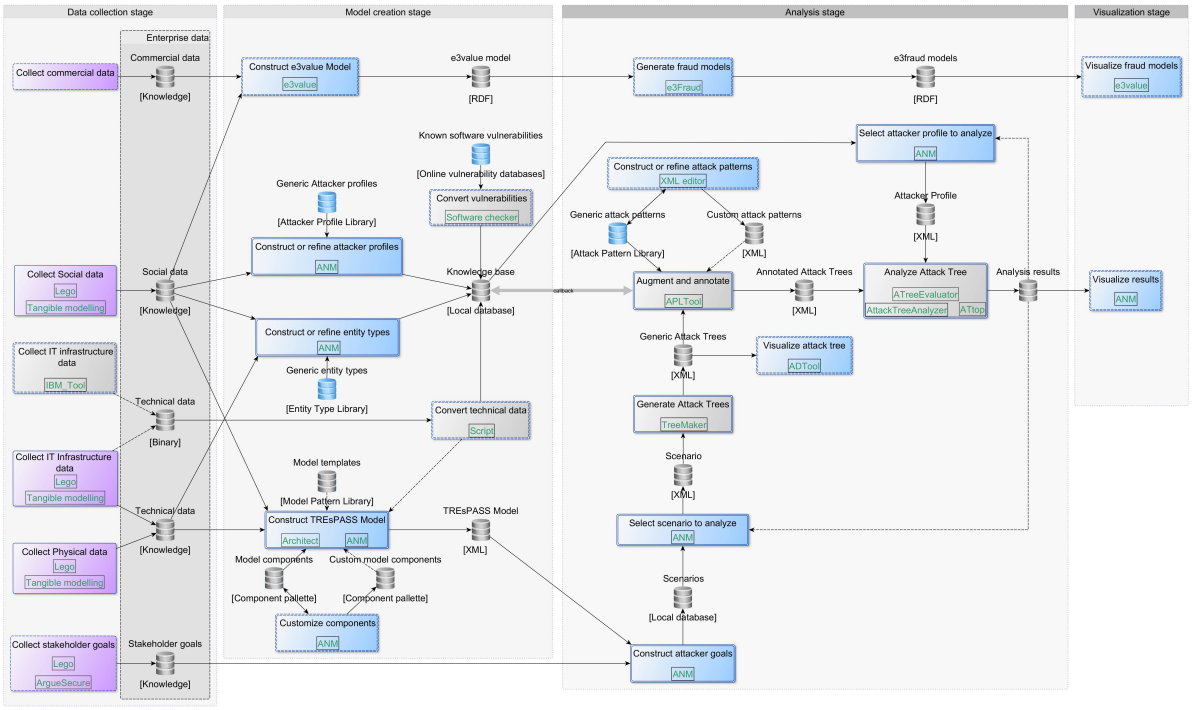
Expressivity

Expressivity refers to the power or potential to affect a viewer. Expressivity is an index not just of the power to evoke through visuality, but also it is an index of receptivity, an index of audience reactions.

B

Overview of TREsPASS integration

The following section presents an overview of TREsPASS workflow as described in the TREsPASS tools handbook (D6.4.2) [197]



Source: TREsPASS project

The **Data collection** stage prepares for analysis and modelling steps, and may require the gathering of one or more of the following kinds of data.

Physical data collection provides knowledge about the physical layout of the organisation including locations, buildings, rooms, doors, windows, etc.

Digital data collection gathers information about the organisation's IT infrastructure.

Social data collection focuses on organisational and individual data, and results in actor profiles containing, e.g. attributes of employees, stakeholders, or potential attackers.

Commercial data collection gathers information required for e3fraud analyses, which focus on potential fraud.

Stakeholder goal collection identifies assets and policies the protection of which is critical to one or more stakeholders.

The **model creation stage** handles the creation of the TREsPASS model and associated actor profiles.

TREsPASS model creation is a key activity resulting in a system model that can be further extended and analysed. The e3value model creation process is complementary to the main TREsPASS model, for cases requiring a more specific financial

focus:

Components customisation (optional) takes place before or during the TRESPASS model creation to create specialised custom model components.

Attacker profile creation creates the attacker profile that the TRESPASS model analysis should consider, based on ready-made attacker profiles.

Defender/target profile creation creates similar profiles for the other actors in the model based on the social data gathered in the social data collection activity.

e3value model creation This interactive activity involves using e3value toolkit2 to create business value models. These models structure the commercial information gathered in the data collection stage in a formal way.

In the analysis stage different analyses are possible depending on the model chosen. The analysis of the TRESPASS model involves these steps:

1. In the **attacker profile selection**, the user selects the attacker profile to use in the analysis.
2. The **attacker goals creation** provides the attack generation with the attacker goals. These can be derived by hand from the stakeholder goals or deduced automatically from the selected attacker profiles.
3. The **scenario selection** selects a scenario, consisting of a single pair of attacker and attacker goal, to run the TRESPASS analysis on.
4. To extend attack trees, **attack pattern creation and sharing** provides libraries with known attack steps. The attack tree generation can only reach a certain level of abstraction, which may not be sufficient for quantitative analyses.
5. **Attack generation** transforms the TRESPASS model to an attack tree.
6. **Attack tree annotation & augmentation** then extends the attack tree with attack patterns and decorates leaf nodes with parameter values from the data collection stage for quantitative analysis.
7. The **attack tree analyses** compute quantitative properties of attacks, e.g. utility for the attacker or success probability of the attack.

The analysis of the **e3value model** is complementary to the core TRESPASS analysis and has only one step:

For the **fraud model generation**, the user needs select an attacker and an interval of expected occurrence rates of the commercial transactions specified by the e3value model. The e3fraud tool then identifies all possible violations of contracts, the loss for actors, and the delta in profit for the other actors.

The **visualisation stage** can be used continuously to provide practitioners with feedback regarding the results of their activities:

1. **Fraud model visualisation** shows the generated attacks as a ranked list of textual descriptions of the attack steps and displays charts showing the profitability for each actor.
2. **Attack tree visualisation** shows the intermediary attack trees.
3. **Attack tree analysis visualisation** visualises analysis results.

C

Semi-structured interview protocol

Following the principle of any semi-structured interviews, the questions raised will not be limited to those listed below. Instead, new questions may stem out of these questions, and therefore they could be seen as pointers leading towards more open questions. As the name suggests, probes are asked when the respondents have difficulty in understanding the respective questions, or when the discussions need to be stimulated due to responses that are rather curtailed. The protocol is given as follows:

1. **Introduction (5 minutes)**

   Thank you for agreeing to meet with me. I am Kevin Joseph Syauta from Delft University of Technology. For my Master thesis, I am speaking with people involved within the TRESPASS project to collect expert opinions. I would like to speak with you about the concept of traceability in risk assessment and how it relates to TRESPASS. What I learn from today will help me in writing my Master thesis about traceability in the TRESPASS model, particularly in building a trace model to support the TRESPASS model. Before we begin, I have to convey that I will not treat your answers as confidential. Your names will be included in my Master thesis. I would like to ask your permission to record this interview for later analysis. However, the transcripts of the recording will not be part of my Master thesis. Do you have any questions about the study?

2. **Traceability (10 minutes)**

   (a) *How would you define traceability in a risk assessment process? What should it consist of?*
   PROBE: You could mention three things that come to your mind when hearing the words "traceability" and "risk".

3. **Traceability in TRESPASS (30 minutes)**

   Now I'd like to take you to delve deeper into the notion of traceability in the context of TRESPASS

   (a) *What do you think are the opportunities for traceability in TRESPASS?*
   PROBE: How do you think traceability feature can bring added value to TRESPASS?

   (b) *What do you think are the necessary requirements to be able to support traceability in TRESPASS? What features should a traceability support for TRESPASS have?*

   (c) *Ideally, how do you think traceability should be added to TRESPASS? What form should it take?*
   PROBE: Should it take the form of a *language extension*, *internal embeddings*, or a *tool support*?

   (d) *Here is a picture of the integration diagram of TRESPASS annotated with possible trace links between its elements (picture is shown). What do you think of these trace links? Do they make sense? Do you think more links need to be added? Do you think certain links should not exist?*

4. **Traceability and dynamic risk assessment (10 minutes)**

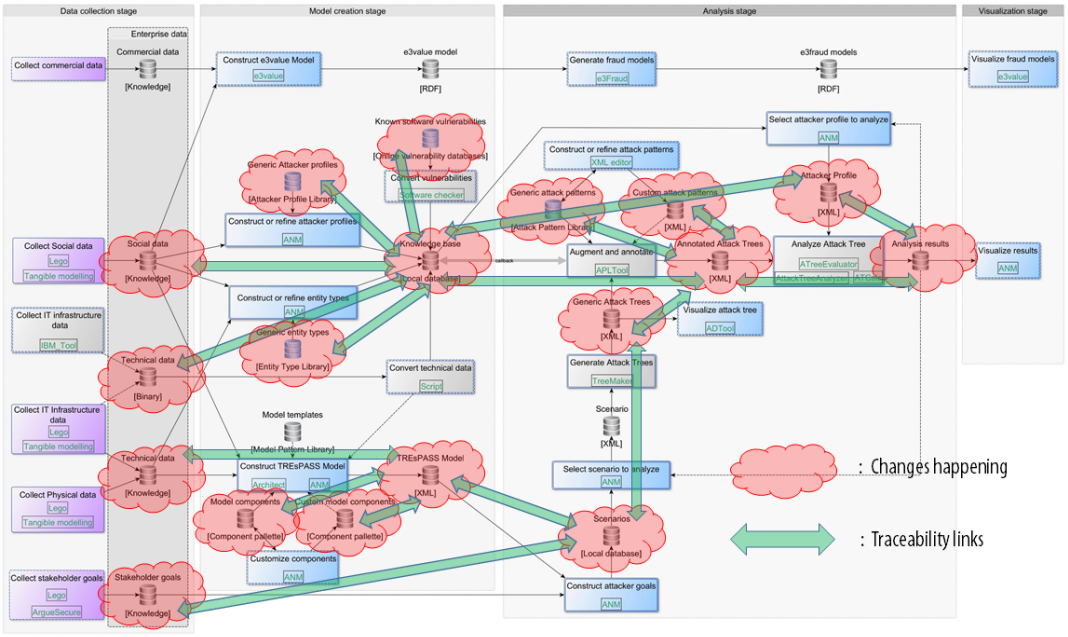
   Now I'd like to hear your opinion on the relation of traceability with dynamic risk assessment.

   (a) *How do you see the role of traceability in facilitating a dynamic risk assessment?*
   PROBE: How do you see traceability could help with keeping the validity of a risk picture after changes happened?

5. **Final thoughts and closing (5 minutes)**

   Those were all the questions I wanted to ask. Do you have any final thoughts about the topics we just discussed (traceability) or TRESPASS project in general that you would like to share?

   Thank you for your time.



Note: Integration diagram is adopted from [190], green arrows are added.

D

Single-case mechanism experiment protocol

1. **Knowledge goal**

   The knowledge goal is to validate the trace model.

2. **Improvement goal**

   The improvement goal is to provide support for traceability in TRE$_S$PASS process according to the definition given in Chapter 4 by developing a trace model with goals formulated in Chapter 5, applied in cloud computing.

3. **Current knowledge**

   The knowledge context consists of the definition of traceability in cyber risk assessment as depicted in Chapter 4 as well as the concepts and integrated process of TRE$_S$PASS.

4. **Conceptual framework**

   The conceptual framework for the trace model is defined in a set of specifications given in Chapter 6. It contains similar complementary concepts to the trace model as defined in the CORAS method, such as *target model* (the TRE$_S$PASS model) and *risk model* (in TRE$_S$PASS, attack trees are used), as well as elements of the TRE$_S$PASS modelling language such as *Actors, Locations, Assets, Relationships*, and *Access Policies*.

5. **Knowledge question**

   The knowledge question is: *Does the trace model work in different contexts of cloud computing environments, namely both in private and outsourced contexts?*

   This question is answered by evaluating the currently devised specifications and see whether they are sufficient for the trace model to work in a different context from the context in which it was designed, or whether they need modifications or even more specifications need to be added.

6. **Intended population**

   The intended population for the trace model is high-tech organisations using cloud computing architectures who seek to measure the socio-technical security posture of their organisation using TRE$_S$PASS. These population elements are similar to one another in the sense that they employ virtualisation technologies in their IT architecture. They are dissimilar to one another in various factors, for instance the complexity of the organisation and the context in which these virtualisation technologies are deployed (in a private or outsourced context).

7. **Object of study**

   Acquisition of object of study

   The validation model is acquired from an existing work on risk assessment in TRE$_S$PASS. The case study should at least be architecturally and contextually different from the case study used to build the trace model. A *disconfirming* case (with regards to the case study used in the design phase) is preferred over a *confirming* one. This way, we can see how the trace model would perform in a different context and provide more learning points.

   We seek to draw *architectural inference* from the validation case. To ensure that this can be achieved, we pay attention to the following criteria: *analysis*, *variation*, and *abstraction*. With regards to analysis, the case should provide enough information about the context of the cloud computing environment. Because our knowledge question is a sensitivity question, a cloud case study with a varying architecture from the design case is chosen. Finally, the level of abstraction is kept to be as similar to the design case as possible to minimise distorting influences.

8. **Treatment design**

   Treatment design here means the *scenarios* that the validation model are exposed to. Similar to how we designed the trace model, we use an existing Use Case from Chapter 6 and combine it with a scenario. To evaluate the specifications

for the static view goals, we use Use Case 1 and 2 with a scenario and similarly we evaluate the specifications for the dynamic view goals with Use Case 3 combined with a different scenario. This is done in a stepwise manner. Another dimension of the treatment design is the example. Examples given are different from the ones given in the design phase. This is intended to see whether we need adjustments to the specifications in different settings. More details are described in Chapter 7.

9. **Measurement design**

   Measurement of the validation is based on the specifications themselves. When these specifications are seen as inadequate for the trace model to perform in the context of the validation case, a proposal for additional specifications or expansion of the existing specifications will be formulated.

10. **Inference design**

    Inferences are drawn in three different ways: *description*, *architectural explanation*, and *generalisation by analogy*. Descriptive inference is mostly presented with a walkthrough of the processes involved in the Use Cases.

    With abductive inference, we foresee the possible explanations of the trace model's behaviour in an outsourced context. These explanations are grounded on situations that commonly occur with outsourced cloud context. Here we only expect to draw *architectural inference*. The other types of inference (*causal inference* and *rational inference*) do not apply to our experiment as we only have one object of study for the experiment (hence no dynamics between different objects of study) and there are no actors involved in the experiment (independent; therefore no reasons to explain actors' rationality).

    Analogic inference is drawn by analysing how (dis)similar the validation case is to the design case. We also analyse the treatments applied and the measurement to see how they contribute to the generalisability of the trace model in different cloud settings.

11. **Research execution**

    The execution phase of the single-case mechanism experiment will be recorded thoroughly, starting from the selection of the target model up to the measurement phase. This includes the process behind the selection of the validation case, developments and decisions within the project that influence the validation process, and the usage of available data.

12. **Data analysis**

    Descriptions: Here we explain the steps involved in data preparation and data management as well as the availability of data necessary for carrying out the experiment. We also briefly address validity of description design, touching base on support for data preparation and interpretation.

    Explanations: Contains explanations of architectural explanations for the observations.

    Generalisations: Contains explanations of the validity of the results in similar cases by referring to analogic inference guidelines, such as the similarity of the object of study and similarity of treatment.

    Answers: Contains answers to the knowledge question (point 5) and briefly addresses contribution to knowledge goal (point 1 and 3) as well as improvement goal (point 2).

E

Concept matrices

This section presents the concept matrices that list prominent themes that show up in literature with respect to traceability. First presented is the concept matrix for traceability in food sector, followed by traceability in requirements engineering and in other domains. We also present the results of our investigation on existing standards and regulations on traceability. We eventually show concept matrices of notions related to traceability, such as data provenance, digital chain of custody, and audit log.

Table E.1: Concept matrix for traceability in food sector

Concepts	Articles																					
	[35]	[217]	[126]	[149]	[78]	[210]	[114]	[127]	[211]	[163]	[2]	[13]	[123]	[77]	[102]	[51]	[146]	[179]	[48]	[122]	[58]	[170]	[150]
Tracing	X		X	X				X	X	X								X			X		
Tracking		X						X	X	X											X		
Traceback system			X				X							X									
Product (food/feed/animal/substance)	X	X	X	X				X		X	X	X		X				X		X	X		
Stages (production/processing/distribution)	X		X	X				X			X			X		X		X			X		
Data collecting/retrieval		X	X	X											X								
Detection		X																					
Labeling		X	X		X	X	X					X										X	
Records							X																
Parameters		X																					
Trace links															X								
Origin/source			X		X	X	X		X		X			X			X					X	
Ownership																						X	
Requirements																						X	
Compliance																						X	
Lifecycle															X								
History			X	X								X									X		
Identity																X							
Map			X																				
Chronology			X																				
Identification			X	X				X							X					X	X		
Control			X				X										X						
Quality			X																		X		
Transparency			X								X									X		X	
Record-keeping		X	X																				
Product properties				X																			
Reporting				X																			
Upstream vs downstream				X																			
Routing				X																			
Tools				X																			
Physical entities				X				X															
Locations				X				X															
Agent				X																			
Codes						X																	
Surveillance							X				X											X	
Monitoring							X														X	X	
Voluntary vs mandatory traceability			X			X	X		X					X									
Internal traceability								X															
Chain traceability								X															
Calibration								X															
Routes								X															
Product traceability vs activities traceability								X															
Traceable Resources Unit (TRU)								X															
Backward traceability								X								X							
Reference code									X														
Safety	X	X	X	X	X	X	X	X	X	X											X	X	
Tracer substance										X													
One step up, one step down													X								X		
Liability														X									
Breadth, depth, precision														X									
Ex-post traceability														X									
Physical traceability															X								
Quality information management															X								
Batch dispersion															X								
Bill of lots-batch distribution															X								
Operation & capacity units															X								
Item observation															X								
Authenticity																	X						
Supplier management																	X						
Transformations																					X		
Conventional traceability																			X				
Geographical traceability																			X				
Genetic traceability																			X				X
Chain of custody																				X			
Credibility																				X			
Information transfer functions																				X			
Farm-to-retail traceability																				X			
"From farm to fork"			X			X	X		X			X	X						X		X		

Table E.1 Concept matrix for traceability in food sector (continued)

Concepts	Articles																						
	[35]	[217]	[126]	[149]	[78]	[210]	[114]	[127]	[211]	[163]	[2]	[13]	[123]	[77]	[102]	[51]	[146]	[179]	[48]	[122]	[58]	[170]	[150]
Producer-traceable																				X			
Processor-traceable																				X			
Retail-traceable																				X			
National origin-traceable																				X			
Classification																					X		
Logistics traceability																					X		
Qualitative traceability																					X		
Aggregated information flow																					X		
Modeling																					X		
Data processing																					X		
Presentation of data																					X		
Static vs dynamic data																					X		
Mandatory vs optional data																					X		
Value-added & value-based marketing																						X	
Crisis management																						X	

Table E.2: Concept matrix for traceability in requirements engineering

Concepts	**Articles**															
	[12]	[145]	[136]	[41]	[178]	[5]	[75]	[119]	[173]	[70]	[42]	[54]	[74]	[228]	[98]	[142]
Tracing		X					X									
Product			X													
Stages (development/specification/deployment/use)										X						
Trace links (satisfaction, rationale, evolution, dependency)	X	X		X	X	X										
Origin										X						
Source code	X	X			X											
Documentation	X															
Stakeholder		X				X										
Stakeholder requirements	X	X	X	X	X		X			X						
Maintenance	X															
Impact analysis	X			X	X											
Changes	X			X		X										
Reuse	X															
Index	X															
Lifecycle						X										
Dependencies				X								X				
Artifacts				X		X										
Relationship						X										
Design		X	X		X											
Implementation			X													
Objects		X														
Low-end users vs high-end users		X														
Map						X										
Backward traceability				X	X	X				X						
Forward traceability				X	X	X				X						
Validity				X		X										
Test-cases				X	X											
Design rationale				X	X											
Gold-plating				X												
Evolutionary change		X		X	X											
Event-based traceability				X												
Traceability path				X												
Event-based traceability				X												
Direct vs indirect dependencies			X													
Requirements-to-design traceability					X											
Requirements-to-source-code-and-test-cases traceability					X											

Table E.2 Concept matrix for traceability in requirements engineering (continued)

Concepts	**Articles**															
	[12]	[145]	[136]	[41]	[178]	[5]	[75]	[119]	[173]	[70]	[42]	[54]	[74]	[228]	[98]	[142]
Requirements-and-design-to-design-rationale traceability					X											
Evolution of requirements and design					X											
Root-cause analysis					X											
Evolutionary tracing					X											
Transformations						X										
Metamodel		X				X										
Believability: Accuracy, Stability, Utility							X									
Discernability							X									
Endurability							X									
Traceability matrix							X									
Active traceability								X								
Requirements-to-object model									X							
Inter-requirement traceability rules									X							
Pre-requirements-specifications traceability										X						
Post-requirements-specifications traceability										X						
In-place traceability											X					
Trace dependency												X				
Information retrievability							X	X	X	X	X		X	X		
Pre-traceability															X	
Post-traceability															X	
Representation information																X
Specification information																X
Agreement information																X

Table E.3: Concept matrix for traceability in other domains

Concepts	**Articles**																				
	Clinical chemistry				*Complex systems*		*Cryptography*				*Geoscience*	*Manufacturing*	*Medical science*	*Metrology*		*Quality management*		*Supply chain*		
	[132]	[131]	[166]	[174]	[53]	[172]	[112]	[18]	[176]	[175]	[55]	[97]	[69]	[219]	[159]	[104]	[153]	[17]	[67]	[209]
Tracing							X	X	X									X		X
Tracking												X					X	X		X
Product																	X			
Stages																	X			
Origin													X	X						
Stakeholder requirements					X						X									
Relationship					X															
Design					X															
Implementation					X															
Identity							X	X												
Map																	X			
Identification													X							
Unforgeability								X												
Exculpability								X												
Coalition-resistance								X												
Measurement data														X	X					
Instrument											X			X						
Chain traceability																		X		
Calibration			X												X					
Traceable Resource Unit (TRU)																			X	
Backward traceability												X								
Forward traceability												X								
Digital fingerprint										X										
Transfer protocol	X		X																	
Chain of materials	X																			
Flow																	X			
Breadth, depth, precision																			X	
Verification															X					
Timestamp													X							
Classification													X							
Active traceability												X								
Passive traceability												X								
Chain of comparisons		X																		
Metrological traceability		X																		
Traceability lot																		X		
Activity																		X		
Responsible actor																		X		
Global identifier																		X		
Actor identifier																		X		
Traceable entity identifier																		X		
SI traceability				X																

Table E.4: Standards and regulations on traceability

Standard/Regulation	Institution	Domain	Definition	Source
IEEE Std. 29148-2011 Systems and software engineering – Life cycle processes – Requirements engineering	IEEE	Requirements engineering	*"A software requirements specification is traceable if (i) the origin of each of its requirements is clear and if (ii) it facilitates the referencing of each requirement in future development or enhancement documentation"*	[70]
GS1 Global Traceability Standard	GS1	Supply chain	*"The ability to track forward the movement through specified stage(s) of the extended supply chain and trace backward the history, application, or location of that which is under consideration"*	[118]
Regulation (EC) No 178/2002 [1]	EU	Food sector	*"The ability to trace and follow a food, feed, food-producing animal or substance intended to be, or expected to be incorporated into a food or feed, through all stages of production, processing and distribution"*	[109]
Regulation (EC) No 1830/2003 [2]	EU	GMOs	*"The ability to trace GMOs and products produced from GMOs at all stages of their placing on the market through the production and distribution chains"*	[137]
International Vocabulary of Metrology – Basic and general concepts and associated terms (VIM)	JCGM (Joint Committee for Guides in Metrology)	Metrology	*"Property of a measurement result whereby the result can be related to a reference through a documented unbroken chain of calibrations, each contributing to the measurement uncertainty"*	[23]
Principles for Traceability/Product Tracing as a Tool within a Food Inspection and Certification System	Codex Alimentarius Commission	Food sector	*"The ability to follow the movement of a food through specified stage(s) of production, processing and distribution"*	[43]

[1] In effect since January 1, 2005
[2] In effect since November 7, 2003

Table E.4 Standards and regulations on traceability (continued)

Standard/Regulation	Institution	Domain	Definition	Source
ISO 17511:2003 (In vitro diagnostic medical devices – Measurement of quantities in biological samples – Metrological traceability of values assigned to calibrators and control materials)	ISO	Metrology	*"Property of the result of a measurement or the value of a calibration material whereby it can be related to stated references through an unbroken chain of comparisons all having stated uncertainties"*	[95]
ISO 9000:2000 (Quality management systems – Fundamentals and vocabulary)	ISO	Quality management systems	*"Ability to trace the history, application or location of that which is under consideration"*	[59]

Table E.5: Concept matrix for traceability in data provenance

Concepts	Articles																											
	[168]	[25]	[29]	[19]	[169]	[129]	[10]	[227]	[158]	[105]	[61]	[130]	[60]	[49]	[36]	[128]	[125]	[96]	[30]	[28]	[83]	[34]	[205]	[226]	[225]	[110]	[161]	[22]
Lineage	X	X		X																								
Pedigree	X																											
Information	X																											
Sources		X																								X		
Derivation		X																										
Derivation history	X	X																										
Data product	X																									X		
Ancestral data product	X																											
Transformation	X																											
Workflows	X	X			X		X	X					X	X											X			
Data quality	X																											
Audit trail	X																											
Replication recipes	X																											
Attribution	X																											
Data-oriented provenance	X																											
Process-oriented provenance	X																											
Granularity	X																											
Annotations	X															X												
Inversion	X																											
Metadata	X																									X		
Metadata model		X																										
Data acquisition and compilation		X																										
Conversions		X																										
Transformations		X																										
Query-based data processing		X																										
Service-based data processing		X																										
Workflow model		X																										
Version management		X																										
Where-provenance			X												X	X	X	X	X	X								X
Why-provenance			X												X	X	X	X	X	X								X
How-provenance															X	X	X											
Location			X																									
Path			X																									
Uncertain database				X																								
Process provenance					X																			X				
Service invocations					X											X												
Globally Unique Identifier (GUID)					X																							
Provenance lifecycle						X																						
Process documentation						X											X											
Documentation of execution						X																						
User-tailored provenance queries						X																						
Chain of ownership						X																						
Context							X																					
Workflow outputs							X																					
Workflow inputs							X																					
Workflow definitions							X																					
Intermediate data results							X																					
Workflow execution							X																					
Internal provenance								X																				
External provenance								X																				
Semantic provenance									X																			
System provenance/workflow provenance									X																			
Two degrees of separation									X																			
Application-level provenance										X																		
Execution provenance										X																		
Workflow templates										X																		
Workflow instances										X																		
Passive monitoring											X																	
Overriding											X																	
Instrumentation											X																	
Artefacts												X																
Processes												X																
Agents												X																
Causal dependency												X	X															
Abstraction													X															
Prospective provenance													X	X														
Retrospective provenance													X	X														
User-defined information													X	X														

Table E.5 Concept matrix for traceability in data provenance (continued)

Concepts	Articles																												
	[168]	[25]	[29]	[19]	[169]	[129]	[10]	[227]	[158]	[105]	[61]	[130]	[60]	[49]	[36]	[128]	[125]	[96]	[30]	[28]	[83]	[34]	[205]	[226]	[225]	[110]	[161]	[22]
Capture mechanism													X															
Representational model													X															
Infrastructure for storage, access, queries													X															
Eager/bookkeeping/annotation approach															X													X
Lazy/non-annotation approach															X													
Logbook																X												
Directed Acyclic Graph (DAG)																X												
Provenance architecture																	X											
Coarse-grain provenance																				X								
Fine-grain provenance																				X								
Provenance of non-answers																					X							
Instance-level provenance																						X						
Dataset-level provenance																						X						
Provenance store																						X						
Provenance factorisation																						X						
Business provenance																							X					
Auditing																							X					
Traceability																							X					
Data provenance																								X				
Organisation provenance																								X				
Knowledge provenance																								X				
Dependencies																									X			
Change-based provenance																											X	

Table E.6: Concept matrix for digital chain of custody

Concepts	**Articles**										
	[27]	[64]	[103]	[81]	[44]	[46]	[20]	[45]	[66]	[16]	[160]
Documentation	X								X		
Maintenance	X								X		
Chronological history	X								X		
Electronic user identification	X								X		
Digital signature	X	X							X		
Automatic audit trail	X								X		
Non-repudiability properties		X									
Digital logs			X								
Audit records				X							
Who, what, where, why, when, how	X				X	X		X	X	X	
Digital Evidence Management Framework (DEMF)					X						
Characteristics						X					
Dynamics						X					
Factors						X					
Institutions						X					
Integrity						X					X
Evidence collection							X				
Evidence preservation							X				
Analysis							X				
Reporting							X				
Data handling	X				X	X		X			
Confidentiality											X
Authenticity											X
Reliability											X
Data provenance											X

Table E.7: Concept matrix for audit log

Concepts	**Articles**																			
	[156]	[208]	[154]	[155]	[18]	[207]	[206]	[26]	[177]	[204]	[157]	[3]	[180]	[224]	[139]	[203]	[50]	[4]	[68]	[106]
Process mining	X	X	X					X		X	X	X				X				
Conformance checking	X		X	X							X	X								
Business alignment	X		X	X																
Process discovery	X		X							X	X	X								
Event log	X		X					X				X				X	X			
Event log	X	X	X																	
Process instance	X	X	X	X		X														
Activity	X	X	X	X																
Timestamp	X	X	X	X		X														
Performer	X	X	X	X																
Event sequences/traces	X		X			X														
Fitness	X		X																	
Structural appropriateness	X		X																	
Behavioural appropriateness	X		X																	
Process perspective		X																		
Organisational perspective		X																		
Case perspective		X																		
Process log		X																		
Forward security					X									X						
Integrity					X															
Chronological order						X														
Correction of data						X														
Unique identifier						X														
Business process							X													
Task							X													
Agent							X													
Case							X			X						X				
Case data							X													
Role							X													
Push mode									X											
Pull mode									X											
Originator										X						X				
Data elements										X										
Data elements										X										
Session													X							
Event													X							
Blind-Aggregate-Forward logging														X						
State-based auditing															X					
Transition-based auditing															X					
F-measure																	X			
Propagation graphs																		X		
Workflow																		X		X

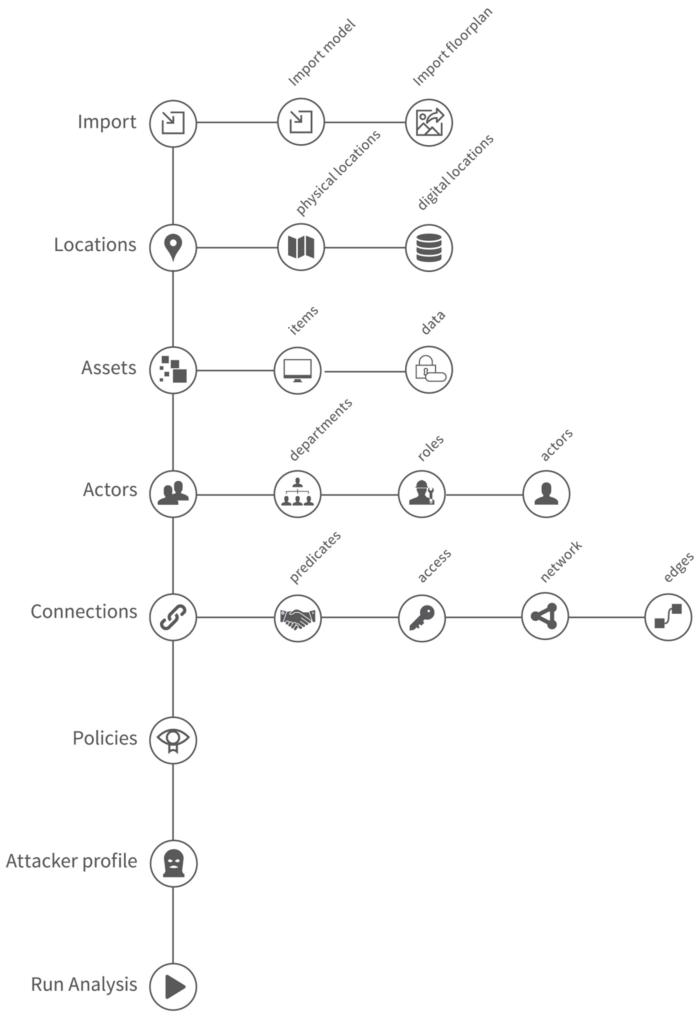
Table E.7 Concept matrix for audit log (continued)

Concepts	**Articles**																			
	[156]	[208]	[154]	[155]	[18]	[207]	[206]	[26]	[177]	[204]	[157]	[3]	[180]	[224]	[139]	[203]	[50]	[4]	[68]	[106]
Information Control Net																				X

F

Sub-wizards of the Attack Navigator Map

This section presents the list of sub-wizards currently supported within the Attack Navigator Map as also presented in D6.3.1 [196]. The sub-wizards appear in the right part of the interface of Attack Navigator Map during model creation.



G

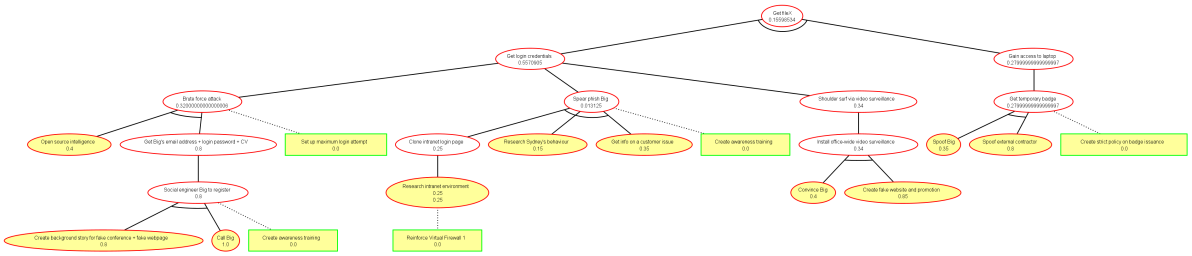
Additional figures and listings used in Chapter 6 and Chapter 7

This Appendix contains additional figures and listings that are required to assist explanation of the Use Cases during the design phase, as well as the validation phase. A clear indication of the context and usage precedes every listing where applicable.

G.1. Chapter 6

G.1.1. Social engineering attack scenario

The attack tree and the language for the social engineering scenario are given as follows:



(Note: Likelihoods are assumed for the purpose of this example)

```
<?xml version='1.0'?>
<adtree>
<node refinement="conjunctive">
<label>Get fileX</label>
<node refinement="disjunctive">
<label>Get login credentials</label>
<node refinement="conjunctive">
<label>Brute force attack</label>
<node refinement="conjunctive">
<label>Open source intelligence</label>
<parameter domainId="ProbSucc1" category="basic">0.4</parameter>
</node>
<node refinement="conjunctive">
<label>Get Big&apos;s email address + login password + CV</label>
<node refinement="conjunctive">
<label>Social engineer Big to register</label>
<node refinement="conjunctive">
<label>Create background story for fake conference + fake webpage</label>
<parameter domainId="ProbSucc1" category="basic">0.8</parameter>
```

```
</node>
<node refinement="conjunctive">
<label>Call Big</label>
<parameter domainId="ProbSucc1" category="basic">0.8</parameter>
</node>
<node refinement="conjunctive" switchRole="yes">
<label>Create awareness training</label>
<parameter domainId="ProbSucc1" category="basic">0.3</parameter>
</node>
</node>
</node>
<node refinement="conjunctive" switchRole="yes">
<label>Set up maximum login attempt</label>
<parameter domainId="ProbSucc1" category="basic">0.8</parameter>
</node>
</node>
<node refinement="conjunctive">
<label>Spear phish Big</label>
<node refinement="conjunctive">
<label>Clone intranet login page</label>
<node refinement="conjunctive">
<label>Research intranet environment</label>
<parameter domainId="ProbSucc1" category="basic">0.25</parameter>
<node refinement="conjunctive" switchRole="yes">
<label>Reinforce Virtual Firewall 1</label>
<parameter domainId="ProbSucc1" category="basic">0.87</parameter>
</node>
</node>
</node>
<node refinement="conjunctive">
<label>Research Sydney&amp;apos;s behaviour</label>
<parameter domainId="ProbSucc1" category="basic">0.15</parameter>
</node>
<node refinement="conjunctive">
<label>Get info on a customer issue</label>
<parameter domainId="ProbSucc1" category="basic">0.35</parameter>
</node>
<node refinement="conjunctive" switchRole="yes">
<label>Create awareness training</label>
<parameter domainId="ProbSucc1" category="basic">0.3</parameter>
</node>
</node>
<node refinement="conjunctive">
<label>Shoulder surf via video surveillance</label>
<node refinement="conjunctive">
<label>Install office-wide video surveillance</label>
<node refinement="conjunctive">
<label>Convince Big</label>
<parameter domainId="ProbSucc1" category="basic">0.4</parameter>
</node>
<node refinement="conjunctive">
<label>Create fake website and promotion</label>
<parameter domainId="ProbSucc1" category="basic">0.85</parameter>
</node>
</node>
</node>
</node>
<node refinement="conjunctive">
<label>Gain access to laptop</label>
<node refinement="conjunctive">
<label>Get temporary badge</label>
```

```
<node refinement="conjunctive">
<label>Spoof Big</label>
<parameter domainId="ProbSucc1" category="basic">0.35</parameter>
</node>
<node refinement="conjunctive">
<label>Spoof external contractor</label>
<parameter domainId="ProbSucc1" category="basic">0.8</parameter>
</node>
<node refinement="conjunctive" switchRole="yes">
<label>Create strict policy on badge issuance</label>
<parameter domainId="ProbSucc1" category="basic">0.4</parameter>
</node>
</node>
</node>
</node>
<domain id="ProbSucc1">
<class>lu.uni.adtool.domains.adtpredefined.ProbSucc</class>
<tool>ADTool2</tool>
</domain>
</adtree>
```

G.2. Example of ArgueSecure arguments

ArgueSecure arguments can also be uploaded to support explanation of modelling activities (CR7 and CR8).

ARGUMENTS													TAGS	
ID	CLAIM		Inference Rules		Assumptions		Facts		Rebuts		Status	Flags	Assets	
#	#	txt	#	txt	#	txt	#	txt		ID(s)	IN/OUT	Transferred / Mitigation	ID	NAME
0	C0	Can lose connectivity to our services	R0	R2,R5, R25 If the external network (internet connection to web server) goes down, there is no connectivity	A0	The cloud provider does not have redundant network uplinks	F0	-			OUT		a1	Company reputation
1	C1	We cannot lose connectivity due to external network going down	R1	Even if the network goes down, there is still connectivity	A1	-	F1	SLA from cloud provider as to guarantee reduntancy	A0		IN	TRANSFERRED??	a2	Customer trust
2	C2	Can lose connectivity to our services		R2, R5, R25 If the internal network (between CSP servers) goes down, there is no connectivit	A2	The cloud privder does not have internal networ redundancy	F2	-			OUT		a3	Employee loyalty and experience

Figure G.1: Snapshot of ArgueSecure arguments for outsourced cloud environment

Source: The TREsPASS project repository

G.3. Model components parameters for ITL

`account`

`id`: account-sydney

`label`: Sydney-admin

`damage_financial`: $ 500,000[1]

`damage_reputation`: Very high

`badge`

`id`: badge-terry

`label`:badge Terry

`cloneability`: Low[2]

`value`: High [3]

[1]Estimation, as compromise of Sydney's account may very likely lead to compromise of ***fileX***.
[2]Assumed, since Terry always carries his badge inside his pocket and rarely leaves it in open space.
[3]Terry's badge grants him full physical access to the whole office.

`damage_financial`: N/A

`damage_reputation`: N/A

`door`

`id`: door-external

`label`: External Door

`size`: Large

`transparency`: None

`burglar_resistance`: High (RC4)

`virtual_machine`

`id`: vm1

`label`: VM1

`value`: N/A

`moveability`: High

`OS`: Linux

`damage_financial`: $ 500,000

`damage_reputation`: Very high

G.4. Updated model of the organisation

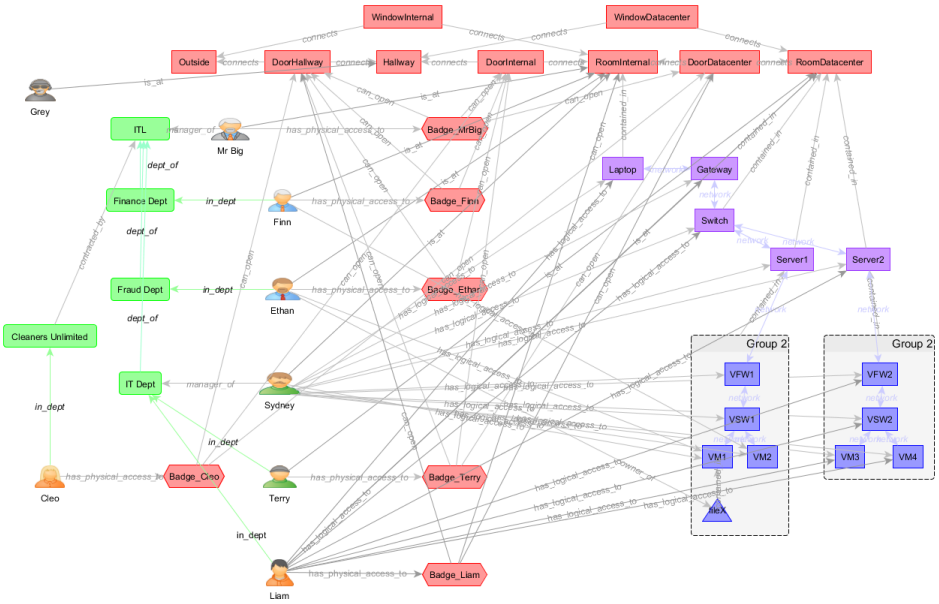
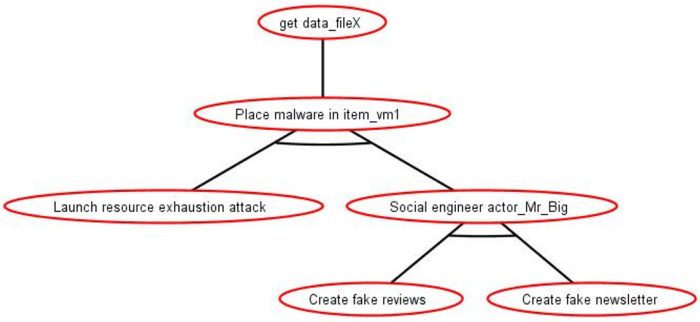


Figure G.2: Updated map of ITL

G.5. Chapter 7

G.5.1. VM image attack scenario

The attack tree for the VM image attack scenario is given as follows:



G.5.2. Attack Tree Analyzer input language

The input language for Attack Tree Analyzer has been rewritten in XML language, providing more explicit explanation regarding the values implied in the nodes. An excerpt for an attack tree for e-voting (courtesy of Cybernetica) is given as follows:

```
        <?xml version="1.0" encoding="UTF-8"?>
        <adtree
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="http://research.cyber.ee/~antonc/adtree.xsd"
        profit="1000000">
        <node refinement="disjunctive">
        <label>Voting tree</label>
        <node refinement="disjunctive">
        <label>Manipulation attack</label>
        <node refinement="disjunctive">
        <label>Attack voter's␣environment</label>
␣␣␣␣␣␣␣␣<node␣refinement="disjunctive">
␣␣␣␣␣␣␣␣<label>Malware</label>
␣␣␣␣␣␣␣␣<node␣refinement="conjunctive">
␣␣␣␣␣␣␣␣<label>Vote␣modifying␣malware</label>
␣␣␣␣␣␣␣␣<node␣refinement="disjunctive">
␣␣␣␣␣␣␣␣<label>Develop␣malware</label>
␣␣␣␣␣␣␣␣<node␣refinement="disjunctive">
␣␣␣␣␣␣␣␣<label>Vote␣changing␣malware␣ML1</label>
␣␣␣␣␣␣␣␣<parameter␣name="cost"␣class="numeric">9600</parameter>
␣␣␣␣␣␣␣␣<parameter␣name="likelihood"␣class="ordinal">V</parameter>
␣␣␣␣␣␣␣␣<parameter␣name="difficulty"␣class="ordinal">M</parameter>
␣␣␣␣␣␣␣␣<parameter␣name="time"␣class="ordinal">HR</parameter>
␣␣␣␣␣␣␣␣</node>
␣␣␣␣␣␣␣␣<node␣refinement="disjunctive">
␣␣␣␣␣␣␣␣<label>Vote␣blocking␣malware␣ML2</label>
␣␣␣␣␣␣␣␣<parameter␣name="cost"␣class="numeric">7680</parameter>
␣␣␣␣␣␣␣␣<parameter␣name="likelihood"␣class="ordinal">L</parameter>
␣␣␣␣␣␣␣␣<parameter␣name="difficulty"␣class="ordinal">M</parameter>
␣␣␣␣␣␣␣␣<parameter␣name="time"␣class="ordinal">HR</parameter>
␣␣␣␣␣␣␣␣</node>
␣␣␣␣␣␣␣␣
```

G.5.3. Timestamp information

Vulnerabilities that can be identified (and taken into account) for the cloud infrastructure are stored in a JSON format file with such declaration:

```
        "meta": {"date-created": "2016-04-07T16:35:35.111797+02:00"
        "epoch-created": 1460039735.111797},
```

The `"meta"` declaration and `"epoch-created"` basically provide the same information that helps users identify the point of time when the vulnerabilities were registered into the knowledge base. The former presents the information in a human-readable version and the latter computer-readable.

Similarly, the file that contains the data for cloud parameters has the capability to record timestamp of creation and version in RDF N3 format:

```
meta:metadata a meta:MetaData ;
meta:creation_date "2016-01-29T17:49:29+01:00" ;
meta:version "0.1"
```

Finally, every instance-specific model persistence has the following metadata for identification:

```
{"meta": {"epoch-created": 1464419896.7915156, "date-modified":
"2016-05-28T07:18:16.791551+00:00", "date-created":
"2016-05-28T07:18:16.791516+00:00", "date-persisted":
"2016-05-28T07:18:16.791688+00:00", "epoch-persisted": 1464419896.791688,
"epoch-modified": 1464419896.791551}}
```

G.6. Unique identification for parameters

The APLTool should be able to generate unique ID for its nodes, under the format of `APLxxx`. The parameters should be connected to this node ID, through XSLT transformation.

```
<node refinement="conjunctive">
<label>Place malware in item_vm1</label>
<id>APL002</id)
<parameter name=\textbf{"cost-success-ratio_APL002"} class="numeric">0</parameter>
<parameter name=\textbf{"cost_APL002"} class="numeric">2000</parameter>
<parameter name=\textbf{"difficulty_APL002"}class="ordinal"H</parameter>
<parameter name=\textbf{"likelihood_APL002"} class="ordinal">V</parameter>
<parameter name=\textbf{"time_APL002"} class="ordinal">D</parameter>

<node refinement="conjunctive">
<label>Launch resource exhaustion attack</label>
<id>APL003</id)
<parameter name=\textbf{"cost-success-ratio_APL003"} class="numeric">0</parameter>
<parameter name=\textbf{"cost_APL003"} class="numeric">500</parameter>
<parameter name=\textbf{"difficulty_APL003"}class="ordinal">M</parameter>
<parameter name=\textbf{"likelihood_APL003"} class="ordinal">V</parameter>
<parameter name=\textbf{"time_APL003"} class="ordinal">HR</parameter>
```

Bibliography

[1] IEEE Guide for Software Requirements Specifications. *IEEE Std 830-1984*, pages 1–26, Feb 1984. doi: 10.1109/IEEESTD.1984.119205.

[2] E Abad, F Palacio, M Nuin, A Gonzalez De Zarate, A Juarros, JM Gómez, and S Marco. RFID smart tag for traceability and cold chain monitoring of foods: Demonstration in an intercontinental fresh fish logistic chain. *Journal of Food Engineering*, 93(4):394–399, 2009.

[3] Rafael Accorsi and Thomas Stocker. On the exploitation of process mining for security audits: the conformance checking case. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, pages 1709–1716. ACM, 2012.

[4] Rafael Accorsi and Claus Wonnemann. Auditing workflow executions against dataflow policies. In *Business Information Systems*, pages 207–217. Springer, 2010.

[5] Neta Aizenbud-Reshef, Brian T Nolan, Julia Rubin, and Yael Shaham-Gafni. Model traceability. *IBM Systems Journal*, 45(3):515–526, 2006.

[6] Christopher J Alberts, Sandra G Behrens, Richard D Pethia, and William R Wilson. Operationally critical threat, asset, and vulnerability evaluation (OCTAVE) framework, Version 1.0. Technical report, DTIC Document, 1999.

[7] Ian Alexander. A taxonomy of stakeholders: Human roles in system development. *International Journal of Technology and Human Interaction*, 1(1):23–59, 2005.

[8] Allianz Global Corporate & Speciality. A Guide to Cyber Risk: Managing the impact of increasing interconnectivity. Research report, Allianz Global Corporate & Speciality, September 2015.

[9] Allianz SE and Allianz Global Corporate & Specialty SE. Allianz Risk Barometer Top Business Risks 2016. Research report, Allianz SE and Allianz Global Corporate & Specialty SE, 2016.

[10] Ilkay Altintas, Oscar Barney, and Efrat Jaeger-Frank. Provenance collection support in the kepler scientific workflow system. In *Provenance and annotation of data*, pages 118–132. Springer, 2006.

[11] M.H. Ammar, M. Benaissa, and H. Chabchoub. 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*, pages 229–234, May 2015. doi: 10.1109/ICAdLT.2015.7136631.

[12] Giuliano Antoniol, Gerardo Canfora, Gerardo Casazza, Andrea De Lucia, and Ettore Merlo. Recovering traceability links between code and documentation. *Software Engineering, IEEE Transactions on*, 28(10):970–983, 2002.

[13] Luis Asensio, Isabel González, Teresa García, and Rosario Martín. Determination of food authenticity by enzyme-linked immunosorbent assay (ELISA). *Food Control*, 19(1):1–8, 2008.

[14] Myo Min Aung and Yoon Seok Chang. Traceability in a food supply chain: Safety and quality perspectives. *Food Control*, 39:172 – 184, 2014. ISSN 0956-7135. doi: http://dx.doi.org/10.1016/j.foodcont.2013.11.007. URL `http://www.sciencedirect.com/science/article/pii/S0956713513005811`.

[15] David L Banks, Jesus M Rios Aliaga, and David Ríos Insua. *Adversarial risk analysis*. CRC Press, 2015.

[16] Hamda Bariki, Mariam Hashmi, and Ibrahim Baggili. Defining a standard for reporting digital evidence items in computer forensic tools. In *Digital Forensics and Cyber Crime*, pages 78–95. Springer, 2010.

[17] Alessio Bechini, Mario GCA Cimino, Francesco Marcelloni, and Andrea Tomasi. Patterns and technologies for enabling supply chain traceability through collaborative e-business. *Information and Software Technology*, 50(4):342–359, 2008.

[18] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *Advances in Cryptology—Eurocrypt 2003*, pages 614–629. Springer, 2003.

[19] Omar Benjelloun, Anish Das Sarma, Alon Halevy, Martin Theobald, and Jennifer Widom. Databases with uncertainty and lineage. *The VLDB Journal*, 17(2):243–264, 2008.

[20] David Bennett. The challenges facing computer forensics investigators in obtaining information from mobile devices for use in criminal investigations. *Information Security Journal: A Global Perspective*, 21(3):159–168, 2012.

[21] F. Bergomi, S. Paul, B. Solhaug, and R. Vignon-Davillier. Beyond traceability: Compared approaches to consistent security risk assessments. In *Availability, Reliability and Security (ARES), 2013 Eighth International Conference on*, pages 814–820, Sept 2013. doi: 10.1109/ARES.2013.109.

[22] Deepavali Bhagwat, Laura Chiticariu, Wang-Chiew Tan, and Gaurav Vijayvargiya. An annotation management system for relational databases. *The VLDB Journal*, 14(4):373–396, 2005.

[23] IEC BIPM, ILAC IFCC, IUPAP IUPAC, and OIML ISO. The international vocabulary of metrology—basic and general concepts and associated terms (vim), 3rd edn. jcgm 200: 2012. *JCGM (Joint Committee for Guides in Metrology)*, 2008.

[24] S. Bleikertz, T. Mastelic, S. Pape, W. Pieters, and T. Dimkov. Defining the cloud battlefield - supporting security assessments by cloud customers. In *Cloud Engineering (IC2E), 2013 IEEE International Conference on*, pages 78–87, March 2013. doi: 10.1109/IC2E.2013.31.

[25] Rajendra Bose and James Frew. Lineage retrieval for scientific data processing: a survey. *ACM Computing Surveys (CSUR)*, 37(1):1–28, 2005.

[26] RP Jagadeesh Chandra Bose and Wil MP van der Aalst. Process diagnostics using trace alignment: opportunities, issues, and challenges. *Information Systems*, 37(2):117–141, 2012.

[27] Halil Ibrahim Bulbul, H Guclu Yavuzcan, and Mesut Ozel. Digital forensics: an analytical crime scene procedure model (ACSPM). *Forensic science international*, 233(1):244–256, 2013.

[28] Peter Buneman and Wang-Chiew Tan. Provenance in databases. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 1171–1173. ACM, 2007.

[29] Peter Buneman, Sanjeev Khanna, and Tan Wang-Chiew. Why and where: A characterization of data provenance. In *Database Theory—ICDT 2001*, pages 316–330. Springer, 2001.

[30] Peter Buneman, Sanjeev Khanna, and Wang-Chiew Tan. On propagation of deletions and annotations through views. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 150–158. ACM, 2002.

[31] Philip Burnard, Paul Gill, K Stewart, E Treasure, and B Chadwick. Analysing and presenting qualitative data. *British dental journal*, 204(8): 429–432, 2008.

[32] Guillermo Horacio Ramirez Caceres and Koji Zettsu. Provenance-based security risk assessment framework. *Journal of information processing*, 22(4):617–625, 2014.

[33] Huseyin Cavusoglu, Birendra Mishra, and Srinivasan Raghunathan. A model for evaluating IT security investments. *Communications of the ACM*, 47(7):87–92, 2004.

[34] Adriane P Chapman, Hosagrahar V Jagadish, and Prakash Ramanan. Efficient provenance storage. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 993–1006. ACM, 2008.

[35] Qasim Chaudhry, Michael Scotter, James Blackburn, Bryony Ross, Alistair Boxall, Laurence Castle, Robert Aitken, and Richard Watkins. Applications and implications of nanotechnologies for the food sector. *Food additives and contaminants*, 25(3):241–258, 2008.

[36] James Cheney, Laura Chiticariu, and Wang-Chiew Tan. *Provenance in databases: Why, how, and where*. Now Publishers Inc, 2009.

[37] Bharat Chhabra and Bhawna Taneja. Cloud computing: Towards risk assessment. In Archana Mantri, Suman Nandi, Gaurav Kumar, and Sandeep Kumar, editors, *High Performance Architecture and Grid Computing*, volume 169 of *Communications in Computer and Information Science*, pages 84–91. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-22576-5. doi: 10.1007/978-3-642-22577-2_12. URL `http://dx.doi.org/10.1007/978-3-642-22577-2_12`.

[38] Chi-An Chih and Yu-Lun Huang. An adjustable risk assessment method for a cloud system. In *Software Quality, Reliability and Security - Companion (QRS-C), 2015 IEEE International Conference on*, pages 115–120, Aug 2015. doi: 10.1109/QRS-C.2015.27.

[39] William R Claycomb and Alex Nicoll. Insider threats to cloud computing: Directions for new research challenges. In *Computer Software and Applications Conference (COMPSAC), 2012 IEEE 36th Annual*, pages 387–394. IEEE, 2012.

[40] Richard Clayton. Anonymity and traceability in cyberspace. Technical Report UCAM-CL-TR-653, University of Cambridge, Computer Laboratory, November 2005. URL `http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-653.pdf`.

[41] Jane Cleland-Huang, Carl K Chang, and Mark Christensen. Event-based traceability for managing evolutionary change. *Software Engineering, IEEE Transactions on*, 29(9):796–810, 2003.

[42] Jane Cleland-Huang, Brian Berenbach, Stephen Clark, Raffaella Settimi, and Eli Romanova. Best practices for automated traceability. *Computer*, (6):27–35, 2007.

[43] Codex Alimentarius Commission et al. Principles for traceability/product tracing as a tool within a food inspection and certification system. *CAC/GL*, 60, 2006.

[44] Jasmin Ćosić and Miroslav Bača. Do we have full control over integrity in digital evidence life cycle? In *Information Technology Interfaces (ITI), 2010 32nd International Conference on*, pages 429–434. IEEE, 2010.

[45] Jasmin Ćosić and Miroslav Bača. (im) proving chain of custody and digital evidence integrity with time stamp. In *MIPRO–Proceedings of the 33rd International Convention*, pages 1226–1230, 2010.

[46] Jasmin Ćosić, Zoran Ćosić, and Miroslav Baća. An ontological approach to study and manage digital chain of custody of digital evidence. *Journal of Information and Organizational Sciences*, 35(1):1–13, 2011.

[47] Sadie Creese, Michael Goldsmith, and Paul Hopkins. Inadequacies of current risk controls for the cloud. In Siani Pearson and George Yee, editors, *Privacy and Security for Cloud Computing*, Computer Communications and Networks, pages 235–255. Springer London, 2013. ISBN 978-1-4471-4188-4. doi: 10.1007/978-1-4471-4189-1_7. URL `http://dx.doi.org/10.1007/978-1-4471-4189-1_7`.

[48] C Dalvit, M De Marchi, and M Cassandro. Genetic traceability of livestock products: A review. *Meat Science*, 77(4):437–449, 2007.

[49] Susan B Davidson and Juliana Freire. Provenance and scientific workflows: challenges and opportunities. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1345–1350. ACM, 2008.

[50] Jochen De Weerdt, Manu De Backer, Jan Vanthienen, and Bart Baesens. A robust f-measure for evaluating discovered process models. In *Computational Intelligence and Data Mining (CIDM), 2011 IEEE Symposium on*, pages 148–155. IEEE, 2011.

[51] David L Dickinson and DeeVon Bailey. Meat traceability: Are us consumers willing to pay for it? *Journal of agricultural and resource economics*, pages 348–364, 2002.

[52] K. Djemame, D. Armstrong, J. Guitart, and M. Macias. A risk assessment framework for cloud computing. *Cloud Computing, IEEE Transactions on*, PP(99):1–1, 2014. ISSN 2168-7161. doi: 10.1109/TCC.2014.2344653.

[53] Michael Edwards and Steven L Howell. A methodology for systems requirements specification and traceability for large real time complex systems. Technical report, DTIC Document, 1991.

[54] Alexander Egyed. A scenario-driven approach to trace dependency analysis. *Software Engineering, IEEE Transactions on*, 29(2):116–132, 2003.

[55] Dara Entekhabi, Eni G Njoku, Paul Houser, Michael Spencer, Terence Doiron, Yunjin Kim, Joel Smith, Ralph Girard, Stephane Belair, Wade Crow, et al. The hydrosphere state (hydros) satellite mission: An earth system pathfinder for global mapping of soil moisture and land freeze/thaw. *Geoscience and Remote Sensing, IEEE Transactions on*, 42(10):2184–2195, 2004.

[56] European Commission. Digital Single Market: Bringing down barriers to unlock online opportunities. `http://ec.europa.eu/priorities/digital-single-market/`, 2015. [Online; accessed January 28, 2016].

[57] Massimo Felici, Valentino Meduri, Bjørnar Solhaug, and Alessandra Tedeschi. Evolutionary risk analysis: expert judgement. In *Computer Safety, Reliability, and Security*, pages 99–112. Springer, 2011.

[58] Dimitris Folinas, Ioannis Manikas, and Basil Manos. Traceability data management for food chains. *British Food Journal*, 108(8):622–633, 2006.

[59] International Organization for Standardization. *Quality Management Systems: Fundamentals and Vocabulary*. International Organization for Standardization, 2000.

[60] Juliana Freire, David Koop, Emanuele Santos, and Cláudio T Silva. Provenance for computational tasks: A survey. *Computing in Science & Engineering*, 10(3):11–21, 2008.

[61] James Frew, Dominic Metzger, and Peter Slaughter. Automatic capture and reconstruction of computational provenance. *Concurrency and Computation: Practice and Experience*, 20(5):485–496, 2008.

[62] Allan Friedman and P.W. Singer. *Cybersecurity and Cyberwar: What everyone needs to know*. Oxford University Press UK, 2014.

[63] Batya Friedman, Peter H Kahn Jr, Alan Borning, and Alina Huldtgren. Value sensitive design and information systems. In *Early engagement and new technologies: Opening up the laboratory*, pages 55–95. Springer, 2013.

[64] Simson L Garfinkel. Providing cryptographic security and evidentiary chain-of-custody with the advanced forensic format, library, and tools. *International Journal of Digital Crime and Forensics (IJDCF)*, 1(1):1–28, 2009.

[65] Shuvanker Ghosh and Gill Hughes. Cloud computing explained. Technical report, The Open Group, 2011.

[66] Giuliano Giova. Improving chain of custody in forensic investigation of electronic digital systems. *International Journal of Computer Science and Network Security*, 11(1):1–9, 2011.

[67] Elise H Golan, Barry Krissoff, Fred Kuchler, Linda Calvin, Kenneth Nelson, Gregory Price, et al. *Traceability in the US food supply: economic theory and industry studies*. US Department of Agriculture, Economic Research Service Washington, DC, 2004.

[68] Mati Golani and Shlomit S Pinter. Generating a process model from a process audit log. In *Business Process Management*, pages 136–151. Springer, 2003.

[69] M Gonzalez-Gross, C Breidenassel, Sonia Gómez-Martínez, M Ferrari, L Beghin, A Spinneker, LE Diaz, G Maiani, A Demailly, J Al-Tahan, et al. Sampling and processing of fresh blood samples within a european multicenter nutritional study: evaluation of biomarker stability during transport and storage. *International Journal of Obesity*, 32:S66–S75, 2008.

[70] Orlena CZ Gotel and Anthony CW Finkelstein. An analysis of the requirements traceability problem. In *Requirements Engineering, 1994., Proceedings of the First International Conference on*, pages 94–101. IEEE, 1994.

[71] B. Grobauer, T. Walloschek, and E. Stocker. Understanding cloud computing vulnerabilities. *Security Privacy, IEEE*, 9(2):50–57, March 2011. ISSN 1540-7993. doi: 10.1109/MSP.2010.115.

[72] Christopher Hadnagy. *Social engineering: The art of human hacking*. John Wiley & Sons, 2010.

[73] William Hau and Rudolph Araujo. Virtualization and risk–key security considerations for your enterprise architecture. *published by McAfee Corporation*, 2007.

[74] Jane Huffman Hayes, Alex Dekhtyar, and James Osborne. Improving requirements tracing via information retrieval. In *Requirements Engineering Conference, 2003. Proceedings. 11th IEEE International*, pages 138–147. IEEE, 2003.

[75] Jane Huffman Hayes, Alex Dekhtyar, and Senthil Karthikeyan Sundaram. Advancing candidate link generation for requirements tracing: The study of methods. *Software Engineering, IEEE Transactions on*, 32(1):4–19, 2006.

[76] Jay Heiser and Mark Nicolett. Assessing the security risks of cloud computing. Research Report G00157782, Gartner, June 2008.

[77] Jill E Hobbs, DeeVon Bailey, David L Dickinson, and Morteza Haghiri. Traceability in the Canadian red meat sector: do consumers care? *Canadian Journal of Agricultural Economics/Revue canadienne d'agroeconomie*, 53(1):47–65, 2005.

[78] LC Hoffman and Eva Wiklund. Game and venison–meat for the modern consumer. *Meat Science*, 74(1):197–208, 2006.

[79] P Hoffman, K Scarfone, and M Souppaya. Guide to security for full virtualization technologies. *National Institute of Standards and Technology (NIST)*, pages 800–125, 2011.

[80] Michael Hogan, Fang Liu, Annie Sokol, and Jin Tong. NIST Cloud Computing Standards Roadmap. *NIST Special Publication*, 35, 2011.

[81] Chet Hosmer. Digital evidence bag. *Communications of the ACM*, 49(2):69–70, 2006.

[82] Siv Hilde Houmb. *Decision Support for Choice of Security Solution: The Aspect-Oriented Risk Driven Development (AORDD) Framework*. PhD thesis, Norwegian University of Science and Technology, Trondheim, 2007.

[83] Jiansheng Huang, Ting Chen, AnHai Doan, and Jeffrey F Naughton. On the provenance of non-answers to queries over extracted data. *Proceedings of the VLDB Endowment*, 1(1):736–747, 2008.

[84] Alina Huldtgren. Design for Values in ICT. In *Handbook of Ethics, Values, and Technological Design*, pages 739–767. Springer, 2015.

[85] Joris Hulstijn and Brigitte Burgemeestre. Design for the values of accountability and transparency. *Handbook of Ethics, Values, and Technological Design*, pages 303–333, 2015.

[86] IEEE. IEEE standard computer dictionary: A compilation of IEEE standard computer glossaries. *IEEE Std 610*, pages 1–217, Jan 1991. doi: 10.1109/IEEESTD.1991.106963.

[87] Joint Task Force Transformation Initiative, Joint Task Force Transformation Initiative, et al. *Managing Information Security Risk: Organization, Mission, and Information System View*. 2011.

[88] Frank Innerhofer-Oberperfler and Ruth Breu. Using an enterprise architecture for it risk management. In *ISSA*, pages 1–12. Citeseer, 2006.

[89] International Organization for Standardization. *ISO 8402: 1994: Quality Management and Quality Assurance-Vocabulary*. International Organization for Standardization, 1994.

[90] Dan Ionita. Current established risk assessment methodologies and tools, July 2013. URL `http://essay.utwente.nl/63830/`.

[91] Dan Ionita, Jan-Willem Bullee, and Roel J Wieringa. Argumentation-based security requirements elicitation: The next round. In *Evolving Security and Privacy Requirements Engineering (ESPRE), 2014 IEEE 1st Workshop on*, pages 7–12. IEEE, 2014.

[92] Dan Ionita, Roel Wieringa, Jan-Willem Bullee, and Alexandr Vasenev. Investigating the usability and utility of tangible modelling of socio-technical architectures. Technical report, May 2015.

[93] ISO. 73: 2009: Risk management vocabulary. *International Organization for Standardization*, 2009.

[94] ISO. ISO/IEC 27005:2011. *Information technology–Security techniques–Information security risk management*, 2011.

[95] EN ISO. 17511: 2003. *vitro diagnostic medical devices–Measurement of quantities in samples of biological origin–Metrological traceability of values assigned to calibrators and control material*.

[96] HV Jagadish, Adriane Chapman, Aaron Elkiss, Magesh Jayapandian, Yunyao Li, Arnab Nandi, and Cong Yu. Making database systems usable. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 13–24. ACM, 2007.

[97] Monique H Jansen-Vullers, Christian A van Dorp, and Adrie JM Beulens. Managing traceability information in manufacture. *International Journal of Information Management*, 23(5):395–413, 2003.

[98] Matthias Jarke. Requirements tracing. *Communications of the ACM*, 41(12):32–36, 1998.

[99] Matthias Jarke, Pericles Loucopoulos, Kalle Lyytinen, John Mylopoulos, and William Robinson. The brave new world of design requirements. *Information Systems*, 36(7):992–1008, 2011.

[100] Jack Jones. An introduction to Factor Analysis of Information Risk (FAIR). *Norwich Journal of Information Assurance*, 2(1):67, 2006.

[101] Vikash Katta and T Stalhane. A conceptual model of traceability for safety systems. *CSDM-Poster Presentation*, pages 1–12, 2010.

[102] Thomas Kelepouris, Katerina Pramatari, and Georgios Doukidis. RFID-enabled traceability in the food supply chain. *Industrial Management & Data Systems*, 107(2):183–200, 2007.

[103] Erin E Kenneally. Digital logs—proof matters. *Digital investigation*, 1(2):94–101, 2004.

[104] Henry M Kim, Mark S Fox, and Michael Gruninger. An ontology of quality for enterprise modelling. In *Enabling Technologies: Infrastructure for Collaborative Enterprises, 1995., Proceedings of the Fourth Workshop on*, pages 105–116. IEEE, 1995.

[105] Jihie Kim, Ewa Deelman, Yolanda Gil, Gaurang Mehta, and Varun Ratnakar. Provenance trails in the wings/pegasus system. *Concurrency and Computation: Practice and Experience*, 20(5):587–597, 2008.

[106] Kwang-Hoon Kim and Clarence A Ellis. Workflow reduction for reachable-path rediscovery in workflow mining. In *Foundations and Novel Approaches in Data Mining*, pages 289–310. Springer, 2006.

[107] Richard Kissel. Glossary of key information security terms. *NIST Interagency Reports NIST IR*, 7298(3), 2013.

[108] Rabia Latif, Haider Abbas, Saïd Assar, and Qasim Ali. Cloud computing risk assessment: A systematic literature review. In James J. (Jong Hyuk) Park, Ivan Stojmenovic, Min Choi, and Fatos Xhafa, editors, *Future Information Technology*, volume 276 of *Lecture Notes in Electrical Engineering*, pages 285–295. Springer Berlin Heidelberg, 2014. ISBN 978-3-642-40860-1. doi: 10.1007/978-3-642-40861-8_42. URL http://dx.doi.org/10.1007/978-3-642-40861-8_42.

[109] General Food Law. Regulation (EC) No. 178/2002 of the European Parliament and of the Council of 28 January 2002, laying down the general principles and requirements of food law, establishing the European Food Safety Authority, and laying down procedures in matters of food safety. *OJ L*, 31(1.2), 2002.

[110] David Leake and Joseph Kendall-Morwick. Towards case-based support for e-science workflow generation by mining provenance. In *Advances in Case-Based Reasoning*, pages 269–283. Springer, 2008.

[111] Ronald Leenes. Framing techno-regulation: An exploration of state and non-state regulation by technology. *Legisprudence*, 5(2):143–169, 2011.

[112] Xiaodong Lin, Xiaoting Sun, Pin-Han Ho, and Xuemin Shen. Gsis: a secure and privacy-preserving protocol for vehicular communications. *Vehicular Technology, IEEE Transactions on*, 56(6):3442–3456, 2007.

[113] Brent J Liu, Zheng Zhou, and HK Huang. A HIPAA-compliant architecture for securing clinical images. *Journal of Digital Imaging*, 19(2): 172–180, 2006.

[114] Maria L Loureiro and Wendy J Umberger. A choice experiment model for beef: What us consumer responses tell us about relative preferences for food safety, country-of-origin labeling and traceability. *Food policy*, 32(4):496–514, 2007.

[115] Mass Soldal Lund, Bjørnar Solhaug, and Ketil Stølen. *Model-driven risk analysis: the CORAS approach*. Springer Science & Business Media, 2010.

[116] Mass Soldal Lund, Bjørnar Solhaug, and Ketil Stølen. Evolution in relation to risk and trust management. *IEEE Computer*, 43(5):49–55, 2010.

[117] Mass Soldal Lund, Bjørnar Solhaug, and Ketil Stølen. Risk analysis of changing and evolving systems using coras. In Alessandro Aldini and Roberto Gorrieri, editors, *Foundations of Security Analysis and Design VI*, volume 6858 of *Lecture Notes in Computer Science*, pages 231–274. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-23081-3. doi: 10.1007/978-3-642-23082-0_9. URL http://dx.doi.org/10.1007/978-3-642-23082-0_9.

[118] GS Mar. Business process and system requirements for full supply chain traceability. *GS1 Global Traceability Standard*, (1.2):2, 2010.

[119] Andrian Marcus and Jonathan I Maletic. Recovering documentation-to-source-code traceability links using latent semantic indexing. In *Software Engineering, 2003. Proceedings. 25th International Conference on*, pages 125–135. IEEE, 2003.

[120] Matt Lobbes. What is dynamic cloud? http://www.thoughtsoncloud.com/2014/02/what-is-dynamic-cloud/, February 28,2014. [Online; accessed January 20, 2016].

[121] McAfee and Centre for Strategic & International Studies. Net losses: Estimating the global cost of cybercrime. *Economic Impact of Cybercrime II*, 2014.

[122] JD McKean. The importance of traceability for public health and consumer protection. *Revue scientifique et technique (International Office of Epizootics)*, 20(2):363–371, 2001.

[123] TA McMeekin, J Baranyi, J Bowman, Paw Dalgaard, M Kirk, T Ross, S Schmid, and MH Zwietering. Information systems in food safety management. *International journal of food microbiology*, 112(3):181–194, 2006.

[124] Peter M. Mell and Timothy Grance. SP 800-145. The NIST Definition of Cloud Computing. Technical report, Gaithersburg, MD, United States, 2011.

[125] Simon Miles, Paul Groth, Miguel Branco, and Luc Moreau. The requirements of using provenance in e-science experiments. *Journal of Grid Computing*, 5(1):1–25, 2007.

[126] M Miraglia, KG Berdal, C Brera, P Corbisier, A Holst-Jensen, EJ Kok, HJP Marvin, H Schimmel, J Rentsch, JPPF Van Rie, et al. Detection and traceability of genetically modified organisms in the food production chain. *Food and Chemical Toxicology*, 42(7):1157–1180, 2004.

[127] T Moe. Perspectives on traceability in food manufacture. *Trends in Food Science & Technology*, 9(5):211 – 214, 1998. ISSN 0924-2244. doi: http://dx.doi.org/10.1016/S0924-2244(98)00037-5. URL http://www.sciencedirect.com/science/article/pii/S0924224498000375.

[128] Luc Moreau. The foundations for provenance on the web. *Foundations and Trends in Web Science*, 2(2–3):99–241, 2010.

[129] Luc Moreau, Paul Groth, Simon Miles, Javier Vazquez-Salceda, John Ibbotson, Sheng Jiang, Steve Munroe, Omer Rana, Andreas Schreiber, Victor Tan, et al. The provenance of electronic data. *Communications of the ACM*, 51(4):52–58, 2008.

[130] Luc Moreau, Ben Clifford, Juliana Freire, Joe Futrelle, Yolanda Gil, Paul Groth, Natalia Kwasnikowska, Simon Miles, Paolo Missier, Jim Myers, et al. The open provenance model core specification (v1. 1). *Future generation computer systems*, 27(6):743–756, 2011.

[131] Mathias M Müller. Implementation of reference systems in laboratory medicine. *Clinical chemistry*, 46(12):1907–1909, 2000.

[132] Gary L Myers, Mary M Kimberly, Parvin P Waymack, S Jay Smith, Gerald R Cooper, and Eric J Sampson. A reference method laboratory network for cholesterol: a model for standardization and improvement of clinical laboratory measurements. *Clinical chemistry*, 46(11): 1762–1772, 2000.

[133] M. Nidd, M. G. Ivanova, C. W. Probst, and A. Tanner. *Tool-based Risk Assessment of Cloud Infrastructures as Socio-Technical Systems*, page 495–517. Elsevier Science Direct. Elsevier, Syngress, Amsterdam, June 2015. doi: http://dx.doi.org/10.1016/B978-0-12-801595-7.00022-7.

[134] Gøran K. Olsen and Jon Oldevik. Scenarios of traceability in model to text transformations. In David H. Akehurst, Régis Vogel, and Richard F. Paige, editors, *Model Driven Architecture- Foundations and Applications*, volume 4530 of *Lecture Notes in Computer Science*, pages 144–156. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-72900-6. doi: 10.1007/978-3-540-72901-3_11. URL `http://dx.doi.org/10.1007/978-3-540-72901-3_11`.

[135] Richard F Paige, Gøran K Olsen, Dimitrios S Kolovos, Steffen Zschaler, and Christopher Power. Building model-driven engineering traceability classifications. 2008.

[136] James D Palmer. Traceability. *Software Requirements Engineering*, pages 364–374, 1997.

[137] Policy Paradigms. Regulation (EC) No. 1830/2003 of the European Parliament and of the Council of 22 September 2003 concerning the traceability and labeling of genetically modified organisms and traceability of food and feed products from genetically modified organisms and amending Directive 2001/18/EC (16). *Official Journal of the European Union*, pages 24–8, 2003.

[138] Stéphane Paul and Olivier Delande. Integrability of design modelling solution. *SecureChange FP7 project deliverable D4.4b*, 4:4b, 2011.

[139] Sean Peisert, Martin Bishop, and Keith Marzullo. Computer forensics in forensis. In *Systematic Approaches to Digital Forensic Engineering, 2008. SADFE'08. Third International Workshop on*, pages 102–122. IEEE, 2008.

[140] Wolter Pieters. Defining" the weakest link" comparative security in complex systems of systems. In *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on*, volume 2, pages 39–44. IEEE, 2013.

[141] Wolter Pieters, Jeroen Barendse, Margaret Ford, Claude PR Heath, Christian W Probst, and Ruud Verbij. The navigation metaphor in security economics. *IEEE Security & Privacy*, 14(3):14–21, 2016.

[142] Klaus Pohl. Pro-art: Enabling requirements pre-traceability. In *Requirements Engineering, 1996., Proceedings of the Second International Conference on*, pages 76–84. IEEE, 1996.

[143] Nayot Poolsappasit, Rinku Dewri, and Indrajit Ray. Dynamic security risk management using bayesian attack graphs. *IEEE Transactions on Dependable and Secure Computing*, 9(1):61–74, 2012.

[144] Christian W. Probst, Jan Willemson, and Wolter Pieters. *Graphical Models for Security: Second International Workshop, GraMSec 2015, Verona, Italy, July 13, 2015, Revised Selected Papers*, chapter The Attack Navigator, pages 1–17. Springer International Publishing, Cham, 2016. ISBN 978-3-319-29968-6. doi: 10.1007/978-3-319-29968-6_1. URL `http://dx.doi.org/10.1007/978-3-319-29968-6_1`.

[145] Balasubramaniam Ramesh and Matthias Jarke. Toward reference models for requirements traceability. *Software Engineering, IEEE Transactions on*, 27(1):58–93, 2001.

[146] Laura T Raynolds. Mainstreaming fair trade coffee: From partnership to traceability. *World development*, 37(6):1083–1093, 2009.

[147] Atle Refsdal, Bjørnar Solhaug, and Ketil Stølen. Risk management. In *Cyber-Risk Management*, pages 9–24. Springer, 2015.

[148] Atle Refsdal, Bjørnar Solhaug, and Ketil Stølen. 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, 2015. ISSN 1433-2779. doi: 10.1007/s10009-014-0351-0. URL `http://dx.doi.org/10.1007/s10009-014-0351-0`.

[149] A. Regattieri, M. Gamberi, and R. Manzini. Traceability of food products: General framework and experimental evidence. *Journal of Food Engineering*, 81(2):347 – 356, 2007. ISSN 0260-8774. doi: http://dx.doi.org/10.1016/j.jfoodeng.2006.10.032. URL `http://www.sciencedirect.com/science/article/pii/S0260877406006893`.

[150] Sanja Risticevic, Eduardo Carasek, and Janusz Pawliszyn. Headspace solid-phase microextraction–gas chromatographic–time-of-flight mass spectrometric methodology for geographical origin verification of coffee. *Analytica chimica acta*, 617(1):72–84, 2008.

[151] Matt Rosenquist. Prioritizing information security risks with threat agent risk assessment. *Intel Corporation White Paper*, 2009.

[152] Ronald S Ross. Guide for conducting risk assessments. *NIST Special Publication*, pages 800–30, 2011.

[153] Aleda V Roth, Andy A Tsay, Madeleine E Pullman, and John V Gray. Unraveling the food supply chain: strategic insights from china and the 2007 recalls*. *Journal of Supply Chain Management*, 44(1):22–39, 2008.

[154] Anne Rozinat and Wil MP van der Aalst. Conformance testing: measuring the fit and appropriateness of event logs and process models. In *Business Process Management Workshops*, pages 163–176. Springer, 2005.

[155] Anne Rozinat and Wil MP van der Aalst. *Decision mining in ProM*. Springer, 2006.

[156] Anne Rozinat and Wil MP van der Aalst. Conformance checking of processes based on monitoring real behavior. *Information Systems*, 33 (1):64–95, 2008.

[157] Anne Rozinat, RS Mans, Minseok Song, and Wil MP van der Aalst. Discovering colored petri nets from event logs. *International Journal on Software Tools for Technology Transfer*, 10(1):57–74, 2008.

[158] Satya S Sahoo, Amit Sheth, and Cory Henson. Semantic provenance for escience: Managing the deluge of scientific data. *Internet Computing, IEEE*, 12(4):46–54, 2008.

[159] E Savio, Leonardo De Chiffre, and R Schmitt. Metrology of freeform shaped parts. *CIRP Annals-Manufacturing Technology*, 56(2):810–835, 2007.

[160] Martin Schäler, Sandro Schulze, and Stefan Kiltz. Database-centric chain-of-custody in biometric forensic systems. In *Biometrics and ID Management*, pages 250–261. Springer, 2011.

[161] Carlos Scheidegger, David Koop, Emanuele Santos, Huy Vo, Steven Callahan, Juliana Freire, and Cláudio Silva. Tackling the provenance challenge one layer at a time. *Concurrency and Computation: Practice and Experience*, 20(5):473–483, 2008.

[162] Klaus Schwab. The Fourth Industrial Revolution: what it means, how to respond. `http://www.weforum.org/agenda/2016/01/the-fourth-industrial-revolution-what-it-means-and-how-to-respond`, January 14, 2016. [Online; accessed February 22, 2016].

[163] F Schwägele. Traceability from a European perspective. *Meat Science*, 71(1):164–173, 2005.

[164] Fredrik Seehusen and Bjørnar Solhaug. Tool-supported risk modeling and analysis of evolving critical infrastructures. In Gerald Quirchmayr, Josef Basl, Ilsun You, Lida Xu, and Edgar Weippl, editors, *Multidisciplinary Research and Practice for Information Systems*, volume 7465 of *Lecture Notes in Computer Science*, pages 562–577. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-32497-0. doi: 10.1007/978-3-642-32498-7_43. URL `http://dx.doi.org/10.1007/978-3-642-32498-7_43`.

[165] A. Seibel. From software traceability to global model management and back again. In *Software Maintenance and Reengineering (CSMR), 2011 15th European Conference on*, pages 381–384, March 2011. doi: 10.1109/CSMR.2011.58.

[166] Christopher T Sempos, Hubert W Vesper, Karen W Phinney, Linda M Thienpont, and Paul M Coates. Vitamin d status as an international issue: national surveys and the problem of standardization. *Scandinavian Journal of Clinical and Laboratory Investigation*, 72(sup243): 32–40, 2012.

[167] Saeed Shafieian, Mohammad Zulkernine, and Anwar Haque. Attacks in public clouds: Can they hinder the rise of the cloud? In *Cloud Computing*, pages 3–22. Springer, 2014.

[168] Yogesh L Simmhan, Beth Plale, and Dennis Gannon. A survey of data provenance in e-science. *ACM Sigmod Record*, 34(3):31–36, 2005.

[169] Yogesh L Simmhan, Beth Plale, and Dennis Gannon. Karma2: Provenance management for data-driven workflows. *Web Services Research for Emerging Applications: Discoveries and Trends: Discoveries and Trends*, page 317, 2010.

[170] GC Smith, JD Tatum, KE Belk, JA Scanga, T Grandin, and JN Sofos. Traceability from a US perspective. *Meat science*, 71(1):174–193, 2005.

[171] Bjørnar Solhaug and Fredrik Seehusen. Model-driven risk analysis of evolving critical infrastructures. *Journal of Ambient Intelligence and Humanized Computing*, 5(2):187–204, 2014. ISSN 1868-5137. doi: 10.1007/s12652-013-0179-6. URL `http://dx.doi.org/10.1007/s12652-013-0179-6`.

[172] Boyeon Song and Chris J Mitchell. RFID authentication protocol for low-cost tags. In *Proceedings of the first ACM conference on Wireless network security*, pages 140–147. ACM, 2008.

[173] George Spanoudakis, Andrea Zisman, Elena Pérez-Minana, and Paul Krause. Rule-based generation of requirements traceability relations. *Journal of Systems and Software*, 72(2):105–127, 2004.

[174] Hedwig CM Stepman, An Vanderroost, Katleen Van Uytfanghe, and Linda M Thienpont. Candidate reference measurement procedures for serum 25-hydroxyvitamin d3 and 25-hydroxyvitamin d2 by using isotope-dilution liquid chromatography–tandem mass spectrometry. *Clinical chemistry*, 57(3):441–448, 2011.

[175] Douglas R Stinson, Tran Van Trung, and Ruizhong Wei. Secure frameproof codes, key distribution patterns, group testing algorithms and related structures. *Journal of Statistical Planning and Inference*, 86(2):595–617, 2000.

[176] Jinyuan Sun, Chi Zhang, Yanchao Zhang, and Yuguang Fang. An identity-based security system for user privacy in vehicular ad hoc networks. *Parallel and Distributed Systems, IEEE Transactions on*, 21(9):1227–1239, 2010.

[177] Smitha Sundareswaran, Anna C Squicciarini, and Dan Lin. Ensuring distributed accountability for data sharing in the cloud. *Dependable and Secure Computing, IEEE Transactions on*, 9(4):556–568, 2012.

[178] Antony Tang, Yan Jin, and Jun Han. A rationale-based architecture model for design traceability and reasoning. *Journal of Systems and Software*, 80(6):918–934, 2007.

[179] Isabel Taverniers, Pieter Windels, Marc Vaïtilingom, Anne Milcamps, Erik Van Bockstaele, Guy Van den Eede, and Marc De Loose. Event-specific plasmid standards and real-time PCR methods for transgenic Bt11, Bt176, and GA21 maize and transgenic GT73 canola. *Journal of agricultural and food chemistry*, 53(8):3041–3052, 2005.

[180] Henry S Teng, Kaihu Chen, and Stephen CY Lu. Security audit trail analysis using inductively generated predictive rules. In *Artificial Intelligence Applications, 1990., Sixth Conference on*, pages 24–29. IEEE, 1990.

[181] The Institute of Risk Management. Cyber Risk Executive Summary. Research report, The Institute of Risk Management, 2014.

[182] The Institute of Risk Management. HPE Security Research Cyber Risk Report 2016. Research report, Hewlett Packard Enterprise, February 2016.

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

[184] The TRE$_S$PASS Project. Initial prototype of the socio-technical security model. Restricted deliverable, October 2013. TRE$_S$PASS Deliverable D1.3.1.

[185] The TRE$_S$PASS Project. Attack generation from socio-technical security models. Restricted deliverable, October 2014. TRE$_S$PASS Deliverable D3.4.1.

[186] The TRE$_S$PASS Project. Current established risk-assessment methods. Public deliverable, October 2014. TRE$_S$PASS Deliverable D5.2.1.

[187] The TRE$_S$PASS Project. Security nightmare 2015 – cloud attack!: Cybercrime social engineering analysis challenge. Flyer, November 2015.

[188] The TRE$_S$PASS Project. Final specifications and requirements for socio-technical security models. Public deliverable, October 2015. TRE$_S$PASS Deliverable D1.1.2.

[189] The TRE$_S$PASS Project. Extensibility of socio-technical security models. Public deliverable, October 2015. TRE$_S$PASS Deliverable D1.3.2.

[190] The TRE$_S$PASS Project. Dynamic features of socio-technical security models. Public deliverable, October 2015. TRE$_S$PASS Deliverable D1.3.3.

[191] The TRE$_S$PASS Project. Final requirements for the data management process. Public deliverable, October 2015. TRE$_S$PASS Deliverable D2.1.2.

[192] The TRE$_S$PASS Project. Data extraction from virtualised infrastructures. Restricted deliverable, October 2015. TRE$_S$PASS Deliverable D2.2.2.

[193] The TRE$_S$PASS Project. TRE$_S$PASS social data and policy extraction techniques. Public deliverable, October 2015. TRE$_S$PASS Deliverable D2.3.2.

[194] The TRE$_S$PASS Project. Best practices for model creation and sharing. Public deliverable, October 2015. TRE$_S$PASS Deliverable D5.3.2.

[195] The TRE$_S$PASS Project. Final refinement of functional requirements. October 2015. TRE$_S$PASS Deliverable D6.2.2.

[196] The TRE$_S$PASS Project. Prototype of the TRE$_S$PASS user interface. Public deliverable, October 2015. TRE$_S$PASS Deliverable D6.3.1.

[197] The TRE$_S$PASS Project. TRE$_S$PASS tools handbook. October 2015. TRE$_S$PASS Deliverable D6.4.2.

[198] The TRE$_S$PASS Project. The TRE$_S$PASS Project. `http://trespass-project.eu/`, 2015. [Online; accessed December 4, 2015].

[199] The TRE$_S$PASS Project. The integrated TRE$_S$PASS process. Restricted deliverable, March 2016. TRE$_S$PASS Deliverable D5.4.2.

[200] The TRE$_S$PASS Project. TRE$_S$PASS information system. Public deliverable, Forthcoming. TRE$_S$PASS Deliverable D2.4.1.

[201] André van Cleeff, Wolter Pieters, and Roel Wieringa. Security implications of virtualization: A literature study. In *Computational Science and Engineering, 2009. CSE'09. International Conference on*, volume 3, pages 353–358. IEEE, 2009.

[202] Jan van den Berg, Jacqueline van Zoggel, Mireille Snels, Mark van Leeuwen, Sergei Boeke, Leo van de Koppen, J.C.A. van der Lubbe, Bibi van den Berg, and Tony de Bos. On (the emergence of) cyber security science and its challenges for cyber security education. *The NATO IST-122 Cyber Security Science and Engineering Symposium*, 2014.

[203] Wil MP van der Aalst. Exploring the CSCW spectrum using process mining. *Advanced Engineering Informatics*, 21(2):191–199, 2007.

[204] Wil MP van der Aalst. Process discovery: capturing the invisible. *Computational Intelligence Magazine, IEEE*, 5(1):28–41, 2010.

[205] Wil MP van der Aalst, Kees M van Hee, Jan Martijn van Werf, and Marc Verdonk. Auditing 2.0: using process mining to support tomorrow's auditor. *Computer*, 43(3):90–93, 2010.

[206] Wil MP van der Aalst, Kees van Hee, Jan Martijn van der Werf, Akhil Kumar, and Marc Verdonk. Conceptual model for online auditing. *Decision Support Systems*, 50(3):636–647, 2011.

[207] Helma van der Linden, Dipak Kalra, Arie Hasman, and Jan Talmon. Inter-organizational future proof EHR systems: a review of the security and privacy related issues. *International journal of medical informatics*, 78(3):141–160, 2009.

[208] Boudewijn F van Dongen, Ana Karla A de Medeiros, HMW Verbeek, AJMM Weijters, and Wil MP van der Aalst. The prom framework: A new era in process mining tool support. In *Applications and Theory of Petri Nets 2005*, pages 444–454. Springer, 2005.

[209] Kees-Jan van Dorp. Tracking and tracing: a structure for development and contemporary practices. *Logistics Information Management*, 15 (1):24–33, 2002.

[210] Wim Verbeke. Agriculture and the food industry in the information age. *European Review of Agricultural Economics*, 32(3):347–368, 2005.

[211] Wim Verbeke and Ronald W Ward. Consumer interest in information cues denoting quality, traceability and origin: An application of ordered probit models to beef labels. *Food Quality and Preference*, 17(6):453–467, 2006.

[212] Verizon Enterprise Solutions. 2016 Data Breach Investigations Report. Research report, Verizon, April 2016.

[213] Piet Verschuren and Rob Hartog. Evaluation in design-oriented research. *Quality and Quantity*, 39(6):733–762, 2005. ISSN 0033-5177. doi: 10.1007/s11135-005-3150-6. URL `http://dx.doi.org/10.1007/s11135-005-3150-6`.

[214] Piet Verschuren, Hans Doorewaard, and MJ Mellion. *Designing a research project*, volume 2. Eleven International Publishing, 2010.

[215] VMWare. Virtualization overview. *White Paper, http://www. vmware. com/pdf/virtualization. pdf*, 2012.

[216] W3C Provenance Incubator Group. What is provenance. `https://www.w3.org/2005/Incubator/prov/wiki/What_Is_Provenance#A_Working_Definition_of_Provenance`, 2010. [Online; accessed March 10, 2016].

[217] Xiaojun Wang and Dong Li. Value added on food traceability: a supply chain management approach. In *Service Operations and Logistics, and Informatics, 2006. SOLI '06. IEEE International Conference on*, pages 493–498, June 2006. doi: 10.1109/SOLI.2006.329074.

[218] Jane Webster and Richard T Watson. Analyzing the past to prepare for the future: Writing a literature review. *MIS quarterly*, 26(2):13–23, 2002.

[219] A Wiedensohler, W Birmili, A Nowak, A Sonntag, K Weinhold, M Merkel, B Wehner, T Tuch, S Pfeifer, M Fiebig, et al. Particle mobility size spectrometers: harmonization of technical standards and data structure to facilitate high quality long-term observations of atmospheric particle number size distributions, atmos. *Meas. Tech. Discuss*, 3:5521–5587, 2010.

[220] Roel Wieringa. An introduction to requirements traceability. In *Faculty of Mathematics and Computer Science, Vrije Universiteit, Tech. Rep. IR-389*, 1995.

[221] Roel J Wieringa. *Design science methodology for information systems and software engineering*. Springer, 2014.

[222] Roel J. Wieringa. Research methods of the TRE$_S$PASS Project, June 2015. Internal document.

[223] World Economic Forum. Global Risks 2016, 11th Edition. Technical report, World Economic Forum, 2016.

[224] Attila A Yavuz and Peng Nin. BAF: An efficient publicly verifiable secure audit logging scheme for distributed systems. In *Computer Security Applications Conference, 2009. ACSAC'09. Annual*, pages 219–228. IEEE, 2009.

[225] Dong Yuan, Yun Yang, Xiao Liu, and Jinjun Chen. On-demand minimum cost benchmarking for intermediate dataset storage in scientific cloud workflow systems. *Journal of Parallel and Distributed Computing*, 71(2):316–332, 2011.

[226] Jun Zhao, Chris Wroe, Carole Goble, Robert Stevens, Dennis Quan, and Mark Greenwood. Using semantic web technologies for representing e-science provenance. In *The Semantic Web–ISWC 2004*, pages 92–106. Springer, 2004.

[227] Jun Zhao, Carole Goble, Robert Stevens, and Daniele Turi. Mining Taverna's semantic web of provenance. *Concurrency and Computation: Practice and Experience*, 20(5):463–472, 2008.

[228] Wei Zhao, Lu Zhang, Yin Liu, Jiasu Sun, and Fuqing Yang. SNIAFL: Towards a static noninteractive approach to feature location. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 15(2):195–226, 2006.