Rest Assured

SECURE DATA PROCESSING IN THE CLOUD

Deliverable D7.3

RestAssured Security and Privacy Engineering

Release 1.0

The research leading to these results has received funding from the European Community's H2020 research and innovation programme under grant agreement n° 731678.

Project Number	:	731678
Project Title	:	RestAssured - Secure Data Processing in the Cloud
Deliverable Type	:	External Report

Title of Deliverable	:	RestAssured Security and Privacy Engineering
Nature of Deliverable	:	External Report
Dissemination Level	:	Public
Contractual Delivery Date	:	31 October 2019
Actual Delivery Date	:	15 December 2019
Contributing WPs	:	WP7
Editor(s)	:	Mike Surridge

Contributors

Name	Organization	Sections
Mike Surridge	IT Innovation	1, 2, 3.2, 3.3, 4.2, 4.3, 5
Toby Wilkinson	IT Innovation	3.2.7, 3.4
Ludger Goeke	UDE	2.2, 3.1, 3.4, 4.1
Ken Meacham	IT Innovation	4.2,4.3
Stefanie Wiegand	IT Innovation	3.2
Nazila Gol Mohammadi	UDE	3.1, 3.4

Document History

Version	Date	Comment
V0.1	15.10.2019	Initial version, copy of D7.2 with placeholders for new content

Contents

List of Figures **7**

List of Tables **9**

1 Introduction **10**

1.1 Report content and structure **10**

1.2 Summary of achievements **11**

2 Risk Assessment Methodology **13**

2.1 A Standards-Based Approach **13**

2.1.1 Relevant standards 13

2.1.2 Current approach and challenges 14

2.1.3 The RestAssured approach 16

2.2 Using RestAssured Tools: SSM and CSAP **24**

2.2.1 Design time risk assessment by the software/service supplier 26

2.2.2 Independent penetration testing of software/services 31

2.2.3 Deployment phase risk assessment by the service operator 32

2.2.4 Penetration testing by the service operator 35

2.3 Risk Levels and Adaptation **35**

2.4 Summary **37**

3 Models **39**

3.1 Modelling the System Context **39**

3.1.1 Problems 39

3.1.2 Description of the ReAs-CSAP 40

3.1.3 The Result of the State of the Art Analysis 44

3.2 Detailed System Modelling **46**

3.2.1 Model Hierarchy 46

3.2.2 Core Model 47

3.2.3 Domain Models 48

3.2.4 System Models 48

3.2.5 Automated inference 49

3.2.6 Threat Models 50

3.2.7 Risk Calculations 57

3.3 The RestAssured Domain Model **64**

3.3.1 Domain model overview 64

3.3.2 Risk evaluation scales . . . 65
3.3.3 Asset classes and relationships . . . 67
3.3.4 Mapping to runtime model concepts . . . 69
3.3.5 Trustworthiness attributes . . . 69
3.3.6 Misbehaviour types . . . 70
3.3.7 Control types . . . 71
3.3.8 Adaptation as a security control mechanism . . . 72
3.3.9 Inference rules . . . 73
3.3.10 Threat models . . . 74
3.3.11 Modelling error checks . . . 82
3.4 Mapping from ReAs-CSAP to Secure System Modeller . . . 84

4 Tools . . . 88
4.1 CSAP-tool . . . 88
4.1.1 The Designer-editor . . . 88
4.1.2 The User-editor . . . 89
4.2 System Security Modeller . . . 90
4.2.1 Architecture . . . 90
4.2.2 User Interface . . . 90
4.2.3 RestAssured enhancements since D7.1 . . . 93
4.3 Run Time Risk Evaluator . . . 99
4.3.1 RRE design . . . 99
4.3.2 Workflow between the adaptation service and RRE . . . 100
4.3.3 Mapping between the runtime model and SSM model variants . . . 103

5 Conclusion and Future Work . . . 105

Bibliography . . . 107

List of Figures

Figure 2.1: Information risk management system - a standardised approach 14

Figure 2.2: Organisational Dependencies in Risk Assessment 15

Figure 2.3: ISO 27005 Figure 2 ([22], Section 6, p. 4): Information Risk Management Process 17

Figure 2.4: Risk assessment and management for a cloud-based application 18

Figure 2.5: RestAssured approach for risk assessment compared to ISO 27005 20

Figure 2.6: Part 1 of methodology for an interacting usage of SSM and CSAP tool 25

Figure 2.7: Part 2 of methodology for an interacting usage of SSM and CSAP tool 26

Figure 2.8: RestAssured approach for risk assessment during design-time compared to ISO 27005 30

Figure 2.9: RestAssured approach for risk assessment during deployment compared to ISO 27005 34

Figure 2.10: Run-time risk assessment in RestAssured 36

Figure 3.1: RestAssured-Cloud System Analysis Pattern (ReAs-CSAP) 40

Figure 3.2: The core model (simplified version) 47

Figure 3.3: Example threat pattern and associated elements 51

Figure 3.4: A primary threat: remote exploit to deny access to a system device 52

Figure 3.5: An secondary threat: loss of availability propagation 53

Figure 3.6: Control strategy for a remote DoS attack: block messages from the malicious attacker 53

Figure 3.7: Control strategy for a remote DoS attack: patch software bugs that can cause a crash 53

Figure 3.8: Interruption of a data flow caused by loss of availability in a process. 54

Figure 3.9: Injecting messages onto a subnet via a network router 55

Figure 3.10: Injecting messages onto a subnet from a connected device 56

Figure 3.11: Risk of no legal basis for processing of personal data 56

Figure 3.12: The Asset class hierarchy 68

Figure 3.13: Risk of illegal cross-border transfer of nationally regulated personal data. 79

Figure 3.14: Implementation of local processing by consent using RestAssured sticky policies 80

Figure 3.15: Implementation of remote processing by consent using RestAssured sticky policies 81

Figure 3.16: Protecting data during processing via RestAssured support for secure enclaves..... 81

Figure 3.17: Detecting servers of non-local data..... 83

Figure 3.18: Detecting data that is served but not used 83

Figure 3.19: Detecting data that is used but not served 83

Figure 3.20: Detecting data that is used but not served 84

Figure 3.21: Detecting data that is used but not served 84

Figure 3.22: Detecting data that is used but not served 84

Figure 4.1: Editor modes of the CSAP-tool..... 88

Figure 4.2: Modification of a Cloud Element in the ReAs-CSAP 89

Figure 4.3: Properties of the Cloud Element of the type Data Protection Platform..... 89

Figure 4.4: Specification of the type of the Cloud 90

Figure 4.5: The System Modeller high-level architecture 91

Figure 4.6: The SSM System Modelling Canvas 92

Figure 4.7: The SSM Threat Catalogue 93

Figure 4.8: The SSM Threat Explorer 94

Figure 4.9: Finding Threats using the SSM Misbehaviour Explorer 95

Figure 4.10: The SSM Report Generator 96

Figure 4.11: Displaying Inferred Relationships 98

Figure 4.12: WP5-WP7 Interfaces..... 101

Figure 4.13: WP5-WP7 Workflow 102

List of Tables

Table 2.1: Possible responses to risk 13

Table 2.2: Risk analysis during software/service design and development 27

Table 2.3: Revaluation of the software/service implementation 31

Table 2.4: Risk analysis at deployment time 32

Table 3.1: Risk calculation symbols and definitions 57

Table 3.2: Domain model developments since RestAssured D7.1 65

Table 3.3: Likelihood and trustworthiness levels in the RestAssured domain model 66

Table 3.4: Misbehaviour impact levels in the RestAssured domain model 66

Table 3.5: Risk levels in the RestAssured domain model 66

Table 3.6: Mapping misbehaviour likelihood and impact to risk level 66

Table 3.7: Process-Data relationships 68

Table 3.8: Threat Coverage 76

Table 3.9: OWASP Top 10 Threats 78

Table 3.10: Mapping from ReAs-CSAP-elements to System Security Modeller-elements 86

1 Introduction

1.1 Report content and structure

This deliverable, D7.3, describes the final outcomes of Work Package 7 (Engineering for Run-Time Data Protection). WP7 comprises asks:

- Task 7.1: Security and privacy by design methodology,
- Task 7.2: Security and privacy threat identification tools,
- Task 7.3: Security and privacy threat mitigation tools.

Task 7.1 covers the overall RestAssured methodology and process, Task 7.2 the development of tools for design-time risk analysis of Cloud-based systems, and Task 7.3 tools for the run-time monitoring of risk levels and modelling of threat mitigation measures. The run-time tools from Task 7.3 consume the design-time system models produced by the tools of Task 7.2, and work in conjunction with the Adaptation components from RestAssured WP5 to address threats to security and privacy in live systems.

This deliverable is organized as follows:

Section 2 describes the overall risk based security and privacy by design methodology, including this refined sequence, its relationship to international standards for information system risk management systems and risk assessment, and the way in which it is supported by the RestAssured design-time and run-time tools.

Section 3 describes in detail the formal models underpinning the two design-time tools. ReAs-CSAP takes a high-level view, modelling the system within a context, using a pattern-based approach that identifies all the stakeholders that relate to the system (Data Subjects, Data Controllers, Cloud Providers etc.). SSM uses a graph-based approach that models the assets of a system and their relationships, and uses a knowledge base of model inference and threat identification rules to generate a threat catalogue and support the calculation of threat likelihoods and the corresponding risk levels.

Section 4 describes the RestAssured tools, including the ReAs-CSAP pattern and the System Security Modeller tools for design-time risk assessment, and the prototype Run-time Risk Evaluator tool including a Risk Dashboard and the intended interactions with the SSM and WP5 Adaptation components. Details are provided of the improvements made in SSM, including new functionality, and usability and performance improvements mainly in response to feedback from validation case studies, but also to allow SSM to be used in the run-time environment. This involves pre-calculating threat catalogues for model variants describing potential run-time system configurations, so a mapping can be found to one of the variants from the WP5 runtime system model, allowing updating of risk level calculations at runtime.

Section 5 sums up the main points of the document and discusses conclusions as well as directions for future work.

This report is separated from the previous WP7 deliverable (D7.2) by only a few months in time. Rather than create a very brief report of updates since D7.2, this report uses the same structure as D7.2 and describes developments since D7.1. The main changes since D7.2 are in the RestAssured Security and Privacy by Design methodology (Section 2) which other than the overview of relevant standards has been completely revised, the addition of some new elements in the security and privacy knowledge base (Section 3.3) providing support for runtime risk assessment, and the evolution of the runtime risk estimation tools (Section 4.3).

1.2 Summary of achievements

In RestAssured we take the view that security and privacy must be designed into systems, but that this on its own is insufficient, and the security of systems once in production must also be continuously monitored and assessed. The security landscape constantly evolves, and new attack techniques are discovered (e.g. the recently discovered Meltdown and Spectre vulnerabilities), and this necessitates continuous assessment of a system's security. In addition, the systems themselves evolve, often in response to changes in the infrastructure the system is deployed upon. This is especially the true in Cloud-based systems where the infrastructure is frequently not owned by the system operator.

Security and privacy issues must therefore be addressed both at design-time, when a system is designed and specified, and at run-time when the system is deployed. In RestAssured we are developing tools for performing design-time risk analysis of systems, and a run-time component that operates in conjunction with the RestAssured Adaptation components to monitor the risk level of a deployed system and assess the risk level of any change to the system proposed by Adaptation.

The tools produced or extended in WP7 are:

- the *System Security Modeller* (SSM), which allows the high level system structure to be captured including interactions between assets, and supports automated risk identification and analysis based on the captured structure.
- the *RestAssured Cloud System Analysis Pattern tool* (ReAs-CSAP), a pattern developed by the project which supports identification of business stakeholders and context related to data protection in the cloud.
- the *Runtime Risk Evaluator tool* (RRE), which supports mapping of information from a run time system model created by adaptation tools from RestAssured WP5 onto an SSM model so its automated risk evaluation can be used at run time.

SSM is based on pre-existing software, and the developments made in RestAssured focus on generating a knowledge base for data protection in the cloud (referred to as the RestAssured Domain Model), and extension or refinements of the software to support its use. It is used at design time to model cloud-based applications, identify risks and security functional requirements, and provide a machine understandable representation allowing comprehensive and precise communication about security assumptions and risks between software/service suppliers and cloud-based application operators.

The ReAs-CSAP pattern is used with a generic, pre-existing CSAP tool to help service operators understand and analyse the context for a planned cloud application including the relationship to its stakeholders, the environment in which it is deployed, and the jurisdictions in which it will operate. The ReAs-CSAP pattern allows service operators to identify information that needs to be added to an SSM model coming from the software/service supplier to allow a pre-deployment security risk assessment and an analysis of regulatory issues that may affect the deployed application.

The RRE is a run-time service, designed to use the SSM deployment model to provide risk assessment updates at run-time. It operates as a service that interfaces with the adaptation tools from RestAssured WP5, allowing the adaptation tools to take security risk levels and regulatory compliance constraints into account in the dynamic, autonomic management of the application to meet other business goals such as cost and performance optimisation.

The main achievements from WP7 are:

- Developing an overall methodology for using these tools to manage risks in cloud-based applications that is aligned with the ISO 27001 and ISO 27005 standards for information security risk management procedures.

- Extending the SSM and CSAP approaches to support this methodology and addressing requirements from RestAssured WP8 validation case studies from Adaptant and OCC.
- Refining the theoretical underpinnings for these two design-time tools, and developing procedures for using them to assess risks prior to application deployment, and to prepare models for run-time risk evaluation.
- Developing a new SSM domain model (knowledge base) for RestAssured, addressing issues raised by the validation studies, and reducing the potential for users to overlook important assets and dependencies.
- Enhancing the two design-time tool prototypes, including extensive performance optimisation of the SSM tool in preparation for its use in support of run-time risk evaluation.
- Creating mappings between the models used by CSAP and SSM, and between SSM models and the runtime system model from RestAssured WP5.
- Extending the SSM knowledge base to remove ambiguities in the mapping to the runtime system model, allowing an automated mapping algorithm to be developed and used to map between WP7 risk models and WP5 system models at runtime.
- Developing an initial prototype of the RRE tool, incorporating this mapping algorithm and using SSM automated risk level calculation algorithms at runtime.
- Engaging with WP8 validation partners and supporting their validation scenarios, providing assistance for the novel risk modelling approach in their applications, and responding to feedback to improve the utility and usability of design-time tools.

The dialogue between WP7 and WP8 has focused mainly on the use of design-time tools, most notably SSM, to create models needed to automate key steps in the overall risk assessment procedure and thus enable the development and use of runtime risk assessment algorithms to support adaptation decisions made by WP5 components.

Adaptant is a micro enterprise providing custom cloud application development and integration for mainly large customer organisations. Developers at Adaptant must consider specific business requirements of their customers, including legal and regulatory constraints, and have focused mainly on these constraints can be enforced at runtime by the RestAssured framework. Collaboration between WP7 and Adaptant has focused less on the RestAssured risk assessment methodology, and more on how to model specific aspects of their application including runtime adaptations for regulatory compliance. OCC is a larger SME providing software, often delivered as a service in the cloud, to customer communities in sectors such as health care and social care. OCC focuses less on the requirements of a single customer and application, and more on challenges that many customers have in common. Collaboration between WP7 and OCC has focused on making sure the overall methodology supports software supply chains, and the design-time (manual) modelling tools are practical and usable and able to add value to software/service suppliers and their customers.

In D7.1 [43], it was assumed that ReAs-CSAP would be used to capture the high-level context, then SSM would be used to add details based on the architecture and dependencies for a specific application. Work with validation partners and especially with OCC led to a reassessment of this simplistic view. In a software/service supply chain, the architecture and dependencies at application level are determined by the supplier of application software and services, taking into account the expected infrastructure which is typically fixed by the (much larger) cloud resource provider. The overall procedure therefore starts with a model of the application architecture created by the software /service supplier using SSM. The service operator then uses CSAP to analyse the context for their specific application of the supplied software/services, and uses its outputs to update the SSM model so it can be used as the starting point for run-time risk evaluation using RRE. In D7.2 the new approach was summarised. The new methodology is now described in detail in Section 2.

2 Risk Assessment Methodology

2.1 A Standards-Based Approach

2.1.1 Relevant standards

The gold standard for managing risks in information systems is ISO/IEC 27001 [20]. This international standard is also a European Normative (EN) standard, having been endorsed by European standards organisations including CEN and CENELEC.

ISO 27001 applies at the level of an organisation, so it is not a standard whose implementation is the focus for RestAssured. However, the RestAssured risk assessment methodology should be usable by organisations that comply (or wish to comply) with ISO 27001.

ISO 27001 specifies the requirements for establishing, implementing and continually improving an information security risk management system within an organisation. It specifies four processes to be carried out by a compliant organisation with respect to an information system (see Figure 2.1):

1. Plan and improve the overall information risk management system;
2. Risk identification, assessment and prioritisation;
3. Implementation of risk reduction measures; and
4. Monitor incidents and evaluate risk management measures.

These four processes are meant to operate as a cycle, so when an incident occurs it is evaluated and lessons learned, as a result of which the overall risk management system can be improved. Steps (1) and (4) are concerned with organisational level aspects of the overall risk management framework. This includes the scope (i.e. which system or collection of systems being addressed), the policies for assessing risks (w.r.t. the purpose of the system), and executive and operational responsibilities.

The steps that are of most interest to RestAssured are steps (2) and (3), each of which has its own associated standard. Step (2) is to identify threats (i.e. sources of risk), and determine what security measures are needed to address them. This is covered by ISO/IEC 27005 [22], which mandates an asset based threat identification procedure, and derivation of risk levels per threat based on their impact and likelihood. Threats can then be prioritised for treatment based on the risk level, using one of the responses in Table 2.1.

Risk response	Description
Risk acceptance	Go ahead with implementation and use of the system despite the associated threat, which is appropriate if the risk level is sufficiently low.
Risk reduction	Introduce security measures to reduce the risk level from the associated threat by making the threat less likely or by mitigating its impact.
Risk transfer	Rely on some other stakeholder to mitigate the risk - this includes relying on them to introduce security measures, or transferring liability by insuring against the risk.
Risk avoidance	Don't use the system features that embody the associated threat, e.g. by replacing or disabling those features, or by just not using the system.

Table 2.1: Possible responses to risk

The level of risk after the introduction of risk reduction or transfer measures is sometimes called the residual risk level. The process we need is one in which risk levels are reduced via these measures to counteract threats, until the residual risk level becomes acceptable. The security requirements for the system

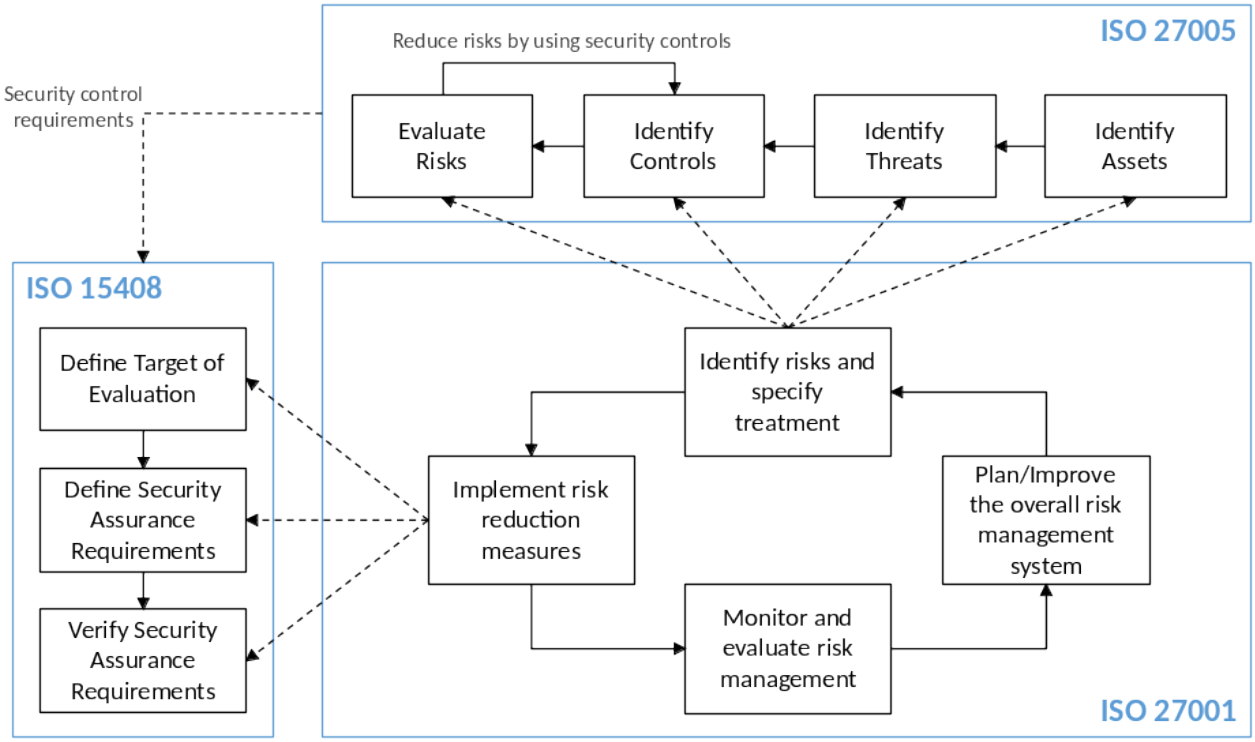

Figure 2.1: Information risk management system - a standardised approach

can then be taken from the measures needed to reach that point. These measures can then be implemented in the system - step (3), and if necessary verified using methods specified by ISO/IEC 15408 [18], also known as Common Criteria.

At this point it is worth mentioning the relationship of RestAssured to several other standards. ISO 27005 and ISO 27001 align the overall risk management process with that of ISO 31000 [17]. ISO 31000 is broader in scope, in that it is not restricted to information security risks, but instead covers risk and opportunity management more generally. This has an obvious relationship to run-time adaptation in RestAssured, where system configuration may evolve for reasons of performance or cost optimisation, and not simply in reponse to information security risks.

Another relevant standard is ISO/IEC 27018 [21]. ISO 27018 augments ISO/IEC 27002 [19] with additional security controls that are applicable when processing personally identifiable information in public clouds. This is obviously relevant to RestAssured, as the RestAssured risk assessment methodology includes selection of controls to treat identified risks, and in particular, the risks addressed include threats to personally identifiable information.

ISO 27001 and related standards are well established, and it is not a goal of RestAssured WP7 to propose changes to them. The focus is to provide tools to support engineering of cloud based applications in compliance with these standards. In particular, we aim to support the continuous analysis of risks, as required by the GDPR, in a manner consistent with ISO 27005. This is in effect impossible without access to automated tools and methods of the kind developed by WP7.

2.1.2 Current approach and challenges

Evidently, each of these standards has a distinct scope and purpose:

- ISO 27001 applies to an organisation: it explains how an organisation should manage information security risks, including a list of control measures that should be considered;

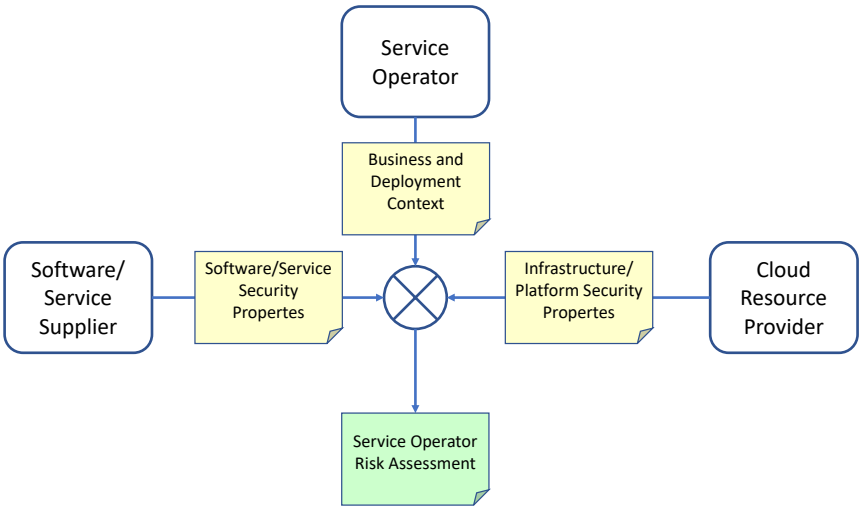

Figure 2.2: Organisational Dependencies in Risk Assessment

- ISO 27005 applies to a (socio-technical) information system in an operational context: it explains how to systematically identify and analyse information security risks using an asset-based approach;
- ISO 15408 applies to a system, product or service known as the Target of Evaluation (TOE): it explains how one can describe and verify the security properties of the TOE.

ISO 27005 provides the link between a technical, system-centric assessment and the organisational, business context. However, in practice, many organisations use an approach based on the concepts from ISO 27001 and 15408 only. Rather than following the procedure from ISO 27005, they instead use ISO 27001 Annex A to analyse risks in terms of security control requirements. One reason for this is that ISO 27001 requires a risk assessment that considers the security control requirements from ISO 27001 Annex A, but it doesn't mandate the use of ISO 27005. Another reason is that the service operator depends on getting input from software/service developers and cloud resource providers, and this is usually expressed in ways that are more easily related to control requirements.

The simplest way to do this is by exchanging information about security properties or controls, as indicated in Figure 2.2, because this information does not depend on the business requirements or context.

Today, the dependency on suppliers of specialised software or services (including companies like RestAssured partners Adaptant and OCC) can normally be resolved in a reasonable way. According to OCC, many service operators are satisfied if they can verify that OCC does comply with ISO 27001, or that OCC's software is subject to independent security testing. Neither of these assurances relates directly to the service operator's own risk analysis, and neither directly helps the service operator with that risk analysis.

In a few cases, service operators do request assurances regarding specific security properties, e.g. from penetration tests for specific threats such as command or query injection attacks. This shows the service operator has carried out a risk analysis and determined which security properties they require. In this situation, OCC can provide specific assurances based on results from independent penetration tests. However, even here the information provided is divorced from the operational context for the service operator's system.

The situation regarding cloud resource provision is less satisfactory. The dominant position of cloud providers such as Amazon allows them to dictate who is responsible for which security aspects, and what

information will be provided to service operators. For example, at present Amazon does have ISO 27001 certification, but some of their services lay outside its terms of reference, so an operator who might be satisfied to use only ISO 27001 certified suppliers can only use a subset of Amazon's services. Amazon does allow penetration tests to be performed on and in their platform, but some types of tests are prohibited along with any tests using some platform elements, including S3.

To summarise, today service operators must conduct their own risk assessment, using information from their software/service suppliers and cloud resource providers. However, this information does not relate directly to the service operator's risk assessment and is likely to be incomplete (at least in relation to cloud resources).

Our goal is to devise a methodology that uses RestAssured tools that makes risk assessment easier for service operators, by:

- providing a model of the context assumed by software/service developers and cloud providers when giving assurances on the security of system and platform components;
- allowing the service operator to modify the context and easily discover the implications for their own risk assessment;
- ensuring the resulting model can be used for automated assessment of system changes, meeting the requirement for continuous risk assessment at run time.

The tools available for doing this include the RestAssured tools CSAP-Tool and SSM for design-time risk analysis, and the RRE tool for automatic update of risk assessments at run-time. CSAP-Tool and SSM together address the first two points above, by helping users determine system composition and context, and providing (in SSM) a way to capture this information in a machine understandable way that can be used for pre-deployment risk assessment, and also by the RRE tool for run-time risk assessment. We have also assumed that penetration tests already used by software/service developers should be included in the methodology, which should be aligned with ISO 27005, establishing the explicit relationship between security measures and the business and operational context.

2.1.3 The RestAssured approach

The overall procedure described by ISO 27001 and ISO 27005 implies an orderly sequence as shown in Figure 2.3:

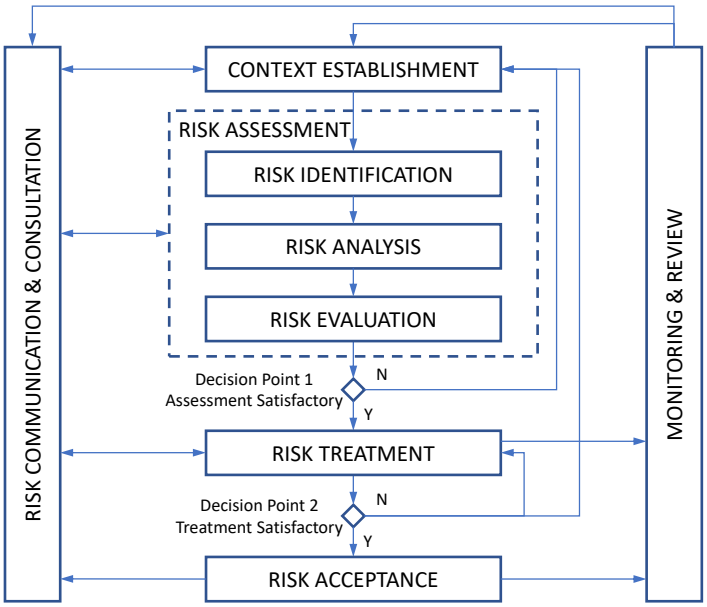

Figure 2.3: ISO 27005 Figure 2 ([22], Section 6, p. 4): Information Risk Management Process

The stages involve the following decisions and analysis:

1. Context establishment: deciding on the risk assessment approach and criteria, and the scope of the risk assessment.
2. Risk assessment consisting of
   (a) Risk identification: identifying system assets and threats with the potential to cause harm to those assets, plus existing security vulnerabilities and controls.
   (b) Risk analysis: determining the likelihood and consequences of threats, and deriving from these the risk level for each threat.
   (c) Risk evaluation: prioritising actions to address risks according to the risk evaluation criteria.
3. Risk treatment: deciding how to treat each risk, which may involve adding security measures to reduce the risk, or simply avoiding, transferring or accepting the risk.
4. Risk acceptance: deciding whether the residual risks are acceptable – if not, one must return to step 1 or 2 and repeat subsequent steps.

This orderly progression makes sense if one considers one stakeholder at a time, but it doesn't really describe the overall process implied by Figure 2.2. Instead, one must consider that each organisation implements the process from Figure 2.3, and that they communicate at points in the process.

Each organisation will of course carry their own risk assessment for their whole organisation. Our concern is to understand the implications for a specific system composed of (cloud hosted) services. Risks to that system are assessed by the service operator, using input from previous risk assessments by their software suppliers and cloud resource providers. The overall process is shown in Figure 2.4.

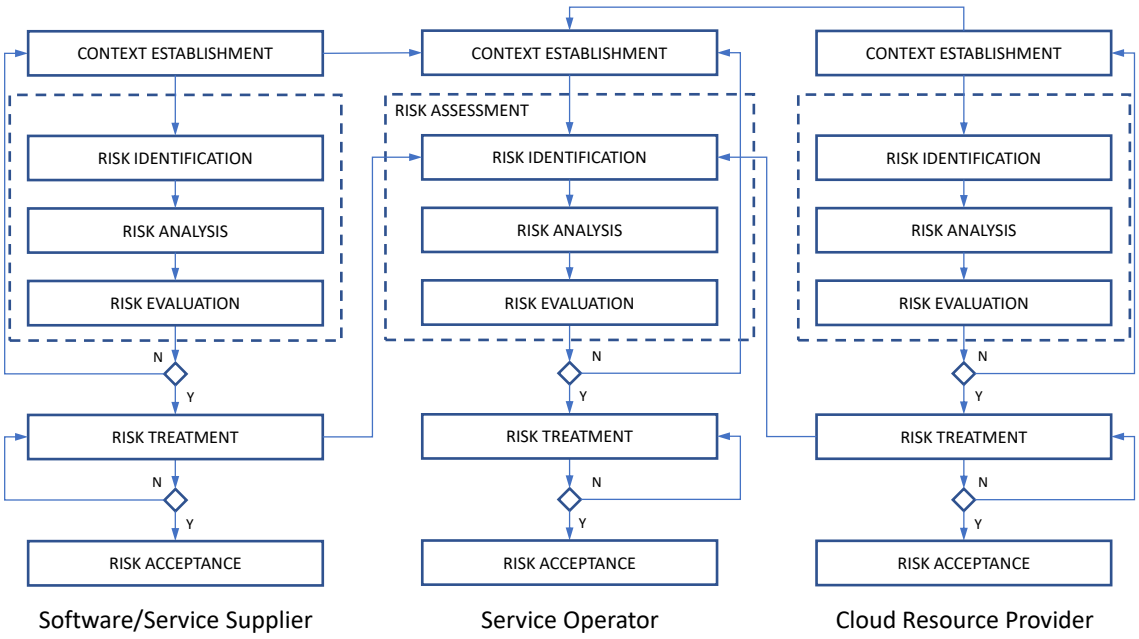

Figure 2.4: Risk assessment and management for a cloud-based application

The key pieces of input required by the service operator concern the risk treatments used by the other two stakeholders, and the context in which the resulting measures are considered adequate. For the reasons noted in Section 2.2, cloud resource providers are rarely prepared to tailor their risk assessment or their security measures and assurances to the needs of an individual service operator, let alone a specific service. It is reasonable to suppose that their analysis was done first and leads to a set of security measures and assurances with which the other stakeholders must work. If the cloud resource provider amends their security stance, the others should update their risk assessment to determine the effect of any changes. The cloud provider's own risk analysis is therefore considered to lie outside the overall RestAssured methodology, but providing a source of inputs.

The overall RestAssured methodology therefore involves the following stages:

1. the software/service supplier conducts a risk assessment for the supplied software/services, taking likely concerns and context of the service operator into account;
2. the software/service supplier performs security testing to provide evidence supporting their risk assessment, typically by employing independent penetration testers;
3. the software/service supplier grants the service operator access to their software/services and an associated risk model encoding relevant information from their risk assessment;
4. the service operator customises the risk model, filing in any missing context and adding their own assessment of trustworthiness for the other two parties;
5. the resulting 'design time' risk model is then used as a starting point to derive possible run-time variants reached via dynamic adaptations;
6. the initial deployment is mapped to one of these variants, and run-time monitoring is then used to continuously update the risk model.

The run-time risk model produced and maintained by this process allows the risk assessment to be dynamically updated whenever there is a significant system change, or prior to inducing a change via a dynamic system adaptation.

2.1.3.1 General guidelines for risk assessment

As already mentioned in Section 2.1.3, the RestAssured risk assessment approach is oriented to the methodology of the ISO 27005 (see Figure 2.3). In this connection, Section 2.1.3 provides an overview over the ISO 27005 methodology and introduces the RestAssured approach.

Because the description of the risk assessment tasks in ISO 27005 is quite abstract, their implementation for concrete services and systems requires additional knowledge. Therefore, this section provides guidelines for an execution of the different risk assessment tasks that are adapted from ISO 27005 within the RestAssured approach by using the RestAssured tool chain. These guidelines consider a usual risk analysis to illustrate how the corresponding tasks from the ISO 27005 methodology are implemented by the RestAssured approach in general. Thus, the understanding regarding the provisions of respective risk assessments during the design-time and deployment of a cloud-based service/system (see Figure 2.4) shall be facilitated. During the description of these provisions in Section 2.2.1 and Section 2.2.3 the specific differences to the explanation in this sections are discussed.

The Figure 2.5 shows a mapping of the risk assessment steps from ISO 27005 to the corresponding implementation within the RestAssured approach. Additionally, for each step of the RestAssured approach the produced output is represented.

In the following, the steps from the RestAssured risk assessment are explained.

(1) Create context model. Before the actual risk assessment, a context establishment has to be performed (see Figure 2.5, ISO 27005 step 1). This context establishment defines the scope and boundaries for the risk analysis. The scope represents the starting point for the identification of the assets of an organisation that are relevant for the risk assessment. For establishing the context, the characteristics regarding the business of an organisation are considered. The corresponding information that presents these characteristics are discussed in Section 3.1.1.

In our approach the context establishment is implemented by creating a context model (see Figure 2.5, RestAssured approach step (1)). To this, we provide a pattern-based model in the form of the ReAs-CSAP (see Section 3.1.2). The ReAs-CSAP defines a pattern that specifies the relevant types of context information for cloud computing services. For the context establishment, a context model is created by instantiating the ReAs-CSAP for the considered cloud computing service. During this instantiation, the pattern elements of appropriate types are instantiated in order to present the context information of the cloud computing service. In this connection, the main business processes are represented in a ReAs-CSAP by instantiated cloud computing services of the levels Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and/or Software as a Service (SaaS) (see Section 3.1.2). The ReAs-CSAP instance also includes the processed data. Furthermore, relevant legislations and regulations in the form of appropriate indirect stakeholders, direct stakeholders of the considered cloud computing service and any relevant locations are identified within a ReAs-CSAP.

Strictly speaking, the data contained in a ReAs-CSAP instance is an asset that is normally identified during the identification of assets (see Figure 2.5, ISO 27005 step 2). Our approach identifies the data already within the context establishment because the types of processed data are necessary for deriving relevant regulations.

A more detailed description of the ReAs-CSAP is provided in Section 3.1.2 .

Tool support for the usage of the ReAs-CSAP is enabled by the *CSAP-tool* (see Section 4.1).

The output of this step is an instance of the ReAs-CSAP.

(2a) Create system model including primary assets. Based on the scope that has been defined by the instantiated ReAs-CSAP in step 1, the assets that are considered within the risk assessment (see Figure 2.5,

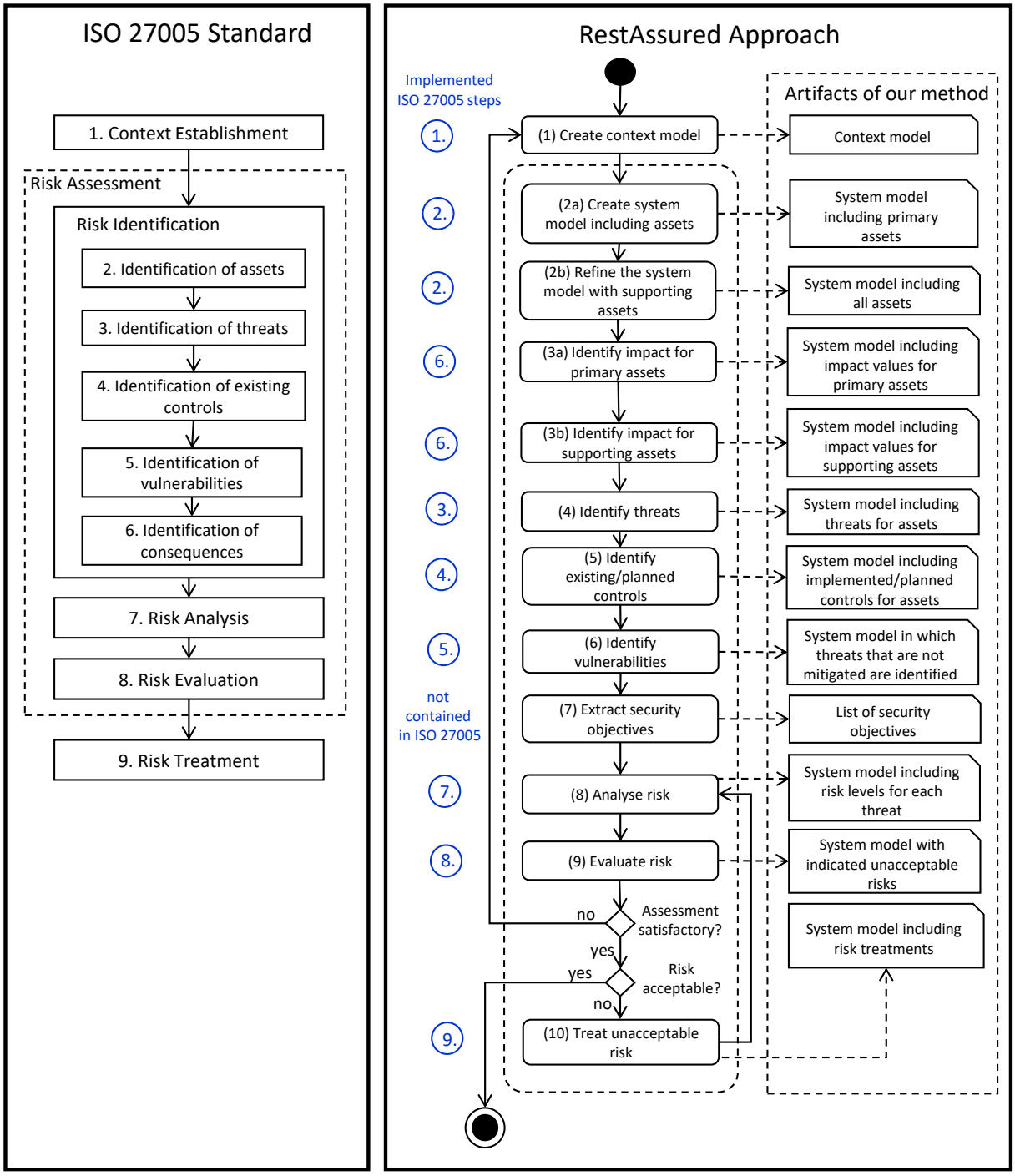

Figure 2.5: RestAssured approach for risk assessment compared to ISO 27005

ISO 27005 step 2) are identified. Regarding assets, the ISO 27005 distinguishes between *primary assets* and *supporting assets*. Primary assets comprise the business processes that are contained in the scope as well as the processed information and data. Supporting assets are those elements on that the primary assets rely (e.g. software, hardware, network).

This step (see Figure 2.5, RestAssured approach step (2a)) relates to the identification of primary assets by creating a system model (see Section 3.2.4). Accordingly, processes and the processed data are modelled in a system model. The modelled processes, are less abstract than the business processes in the ReAs-CSAP instance from the context establishment. The system model defines different types of processes that represent running applications that may process, store or exchange data (see also Section 3.3.3). Therefore, a process in the system model contains always a supporting asset in the form of a software. This software has to be taken into account when a process is considered during the risk assessment. Relations between different processes are also modelled in a system model.

Regarding the data, the system model distinguishes between data with normal protection needs and sensitive data that requires high protection needs. The data shall accord to the data in the ReAs-CSAP instance. Jurisdictions for the data processing are also modelled in a system model. These jurisdictions shall correspond to identified legislations in the ReAs-CSAP instance.

For the creation of a system model we provide tool support by the *System Security Modeller* (see Section 4.2).

The output of this step is a system model that contains primary assets and jurisdictions for the data processing.

The identification of supporting assets is described in the following step 2b.

(2b) Refine the system model with supporting assets. In this step (see Figure 2.5, RestAssured approach step (2b)), the supporting assets for the primary assets from step 2a are identified (see Figure 2.5, ISO 27005 step 2). The RestAssured approach considers the following types of supporting assets that are specified in ISO 27005:

- software;
- hardware;
- networks and their components;
- personnel; and
- sites (cf. [22], Annex B.1.1, p. 28).

Our approach extends theses types of supporting assets by public and private spaces (see Section 3.3.3).

The identified supporting assets are included into the system model that has been created during step 2a. Within the system model, hardware, networks and network components are represented by different types of network assets. For a network asset associations to primary assets in the form of hosted processes and processed, stored and/or exchanged data are identified. Associations that represent connections to or interactions with other network assets are also modelled, and are used by the System Security Modeller to determine attack paths and secondary effect cascades during the later identification of threats for the identified assets in step 4.

Associations to relevant jurisdictions are also taken into account.

Sites are included by a space of the type *data center* that in turn provides network assets.

Personnel is represented by stakeholders of the type *adult*.

For a better overview, stakeholders in person of users are also included into the system model.

The output of this step is the system model that includes all primary and supporting assets.

(3a) Identify impact for primary assets. Within the ISO 27005, the identification of consequences assesses the impact if an asset gets compromised by a threat (see Figure 2.5, ISO 27005 step 6). In this step (see Figure 2.5, RestAssured approach step (3a)) the impact for primary assets is determined by using the System Security Modeller. The RestAssured knowledge base used by the System Security Modeller contains predefined default values for the impact level of each type of compromise and asset type. The default levels are mostly quite low except for sensitive data classes, because in practice this is appropriate for secondary assets, and they can be of any type apart from data. The user has to check and if necessary override the default levels for primary assets only, since the impact of secondary asset compromises will be automatically determined by SSM from the attack paths and secondary effect cascades implied by asset associations.

The output of this step is the system model in which the primary assets have an impact value.

(3b) Identify impact for supporting assets. This step (see Figure 2.5, RestAssured approach step (3b)) determines the impact values for supporting assets by using the System Security Modeller. As in step 3a, the impact values are assigned automatically.

The output of this step is the system model in which the supporting assets have an impact value.

(4) Identify threats. In this step, possible threats to the assets in the system model are identified (see Figure 2.5, ISO 27005 step 3). In our approach, the corresponding step (see Figure 2.5, RestAssured step (4)) is supported by the System Security Modeller. To this, it provides a catalogue of threats that refer to the different types of assets. This catalogue is generated using the threat identification rules incorporated into the RestAssured knowledge base. If new types of attacks are discovered, which are not merely the old attacks exploiting some new programming error, the knowledge base must be updated. One benefit of the RestAssured approch is that the SSM model can then easily be reanalysed: one must upload the updated knowledge base to the SSM service used, and simply re-run the analysis steps to find out if the new type of threat is relevant, and whether it makes a significant difference to the attack paths and secondary effect cascades or leads to new security requirements.

The threats are assigned to the appropriate assets automatically by executing the System Security Modeller functions to validate and populate the system model.

The output of this step is the system model in which the relevant threats are associated to each contained assets.

(5) Identify existing/planned controls. Controls are measures that have been or are planned to be realized for assets to decrease the likelihood that an asset gets compromised by certain threats. Thus, the identification of existing/planned controls is a part of the risk assessment (see Figure 2.5, ISO 27005 step 4). The corresponding step in our approach (see Figure 2.5, RestAssured approach step (5)) is supported by the System Security Modeller. During the model validation stage, SSM generates a catalogue of possible security measures (controls) associated with each system asset. The user can then select controls in the model to indicate which they intend to use to block or mitigate threats and manage risk levels within acceptable limits. SSM finds combinations of these controls (known as control strategies) that could address threats. Where all the controls needed for a control strategy are selected, the effect is to reduces the likelihood of a threat causing the expected consequences. Each control strategy has a strength parameter which determines the maximum residual threat likelihood when the control strategy is selected. See Section 3.2.7.

If new security measures are developed to protect certain types of assets, these should be added to the RestAssured knowledge base, and the system model revalidated to find out if and where they could be used. Again, once the model exists, it is easy to revalidate.

The output of this step is the system model in which the appropriate implemented and/or planned controls are associated to each contained asset.

(6) Identify vulnerabilities. The identification of vulnerabilities (see Figure 2.5, ISO 27005 step 5) of assets is implemented implicitly in our approach (see Figure 2.5, RestAssured approach step (6)). In this connection, a vulnerability of an asset is implied if a threat to an asset is not addressed by an appropriate

control.

(7) Extract security objectives. This step considers the extraction of security objectives (see Figure 2.5, RestAssured approach step (7)) for the considered service/system. It is not adapted from the ISO 27005. The security objectives are specified for the assets under consideration of the identified threats. Security objectives will be used in the specific provisions of our approach (e.g. for the identification of additional controls during design-time). Additionally, security objectives have a significance outside the risk assessment. Thus, they can be used for formulating a designated level of security for a service/system to third parties (e.g. cloud computing providers).

The output of this step is a list of security objectives.

(8) Analyse risk. In the risk analysis (see Figure 2.5, ISO 27005 step 8), for any identified threat regarding each asset the risk level is assessed. A risk level is a quantitative value that represents the risk that an asset gets compromised by the considered threat. It is assessed based on values that have been identified during the previous steps of the risk identification (see Figure 2.5).

The implementation of the risk analysis in our approach (see Figure 2.5, RestAssured approach step (8)) is supported by the System Security Modeller.

Within our approach, the determination of risk levels is performed automatically by executing the appropriate functionality supplied by the System Security Modeller. Risk levels are determined for any threat to each asset. A risk level is determined based on the likelihood that a threat occurs and has the expected consequences, in the presence of selected controls, and the impact of these consequences taking account of attack paths and secondary effects.

The output of this step is the system model in which risk levels are assigned to the threats to each asset.

(9) Evaluate risk. During the risk evaluation (see Figure 2.5, ISO 27005 step 8), it is evaluated if the determined risk levels from the risk analysis are acceptable referring to the defined risk criteria.

The SSM calculates risk levels, but does not automatically decide whether they are acceptable. The RestAssured knowledge base uses a five-point scale to measure the risk level, (see Section 3.3.2), and we suggest that on this scale a Medium risk level is only acceptable for a short time, and normally the worst case risk level from any threat should be Low or Very Low. However, this depends on the risk appetite of the user, who must make a decision after step (9), as shown in Figure 2.5. If any the level of any risk is too high, it must be treated in a further step.

The output of the risk evaluation is the system model in which unacceptable risk levels are indicated.

After the risk evaluation, it has to be evaluated if the previous results of the risk assessment are meaningful (see Figure 2.5). If this is not the case, the scope of the risk assessment has to be adjusted and the steps of the risk assessment have to be repeated. If the results are meaningful, the approach continues with the treatment of unacceptable risks.

(10) Treat unacceptable risk. The risk treatment (see Figure 2.5, ISO 27005 step 9) includes all risk levels, which have been determined during the risk evaluation, that are not acceptable. In our approach (see Figure 2.5, RestAssured approach step (10)), additional controls for the corresponding asset are selected with the objective to reduce the risk level. It is also possible to use other risk treatments, such as avoiding risk (by removing assets and features from the system model). Outsourcing the risk is also possible, but that is not supported directly by the System Security Modeller.

At this point, security controls are selected at each asset with the goal of reducing the level of unacceptable risk. The System Security Modeller provides information about combinations of controls (if any exist) that would counter each threat, so assisting in the choice of which controls should be used. After selection of these additional controls, the steps for the risk analysis and risk evaluation for the concerned assets are repeated to examine if risk treatment has been resulted in acceptable risks.

If this is not the case for certain assets, these assets must undergo a further risk treatment and so on.

2.2 Using RestAssured Tools: SSM and CSAP

This section discusses in more detail how the different stakeholders described in 2.1 can use the *Secure System Modeller (SSM)* and the *Cloud System Analysis Pattern (CSAP)* tools developed in the RestAssured project.

As indicated in Figure 2.4, the service operator requires input from the cloud resource provider and the software/service supplier. We envisage that:

- The cloud operator (assumed to be a very large IT company) will define their offer and provide information about their security measures and responsibilities in their ISO 27001 Statement of Applicability (if any) and the user manual and terms and conditions for using their resources.
- The software/service supplier will use information from the cloud operator to perform their own analysis of the security risks and requirements associated with their software, using SSM to perform the analysis, and thereby producing a machine understandable model of their analysis which can be provided to the service operator.
- The service operator will use CSAP to analyse their intended deployment including business and operational context, then add this information to the SSM model from their software/service provider, producing an SSM model that can be used for their final pre-deployment risk analysis, and subsequently for run-time risk assessment.

This approach for implementing ISO 27005 provides two clear benefits over the normal approach.

The first benefit is that the software/service supplier takes on the first stage of the risk assessment *on behalf of* the service operator. This is something suppliers already do for their customers if they want to be successful, as it is only by understanding a customer's point of view and addressing some of their needs can a supplier deliver value. As previously described, software/service suppliers do typically use penetration testing to allay security concerns and in some cases provide more specific input about security measures incorporated into the service implementation. In the RestAssured methodology this is taken a step further. The software/service supplier contributes directly to the service operator's own risk assessment, providing information about the security properties of the software/service, and also the supplier's assumptions about how it will be used. This is shown in Figure 2.6, and explained in more detail in Section 2.2.1.

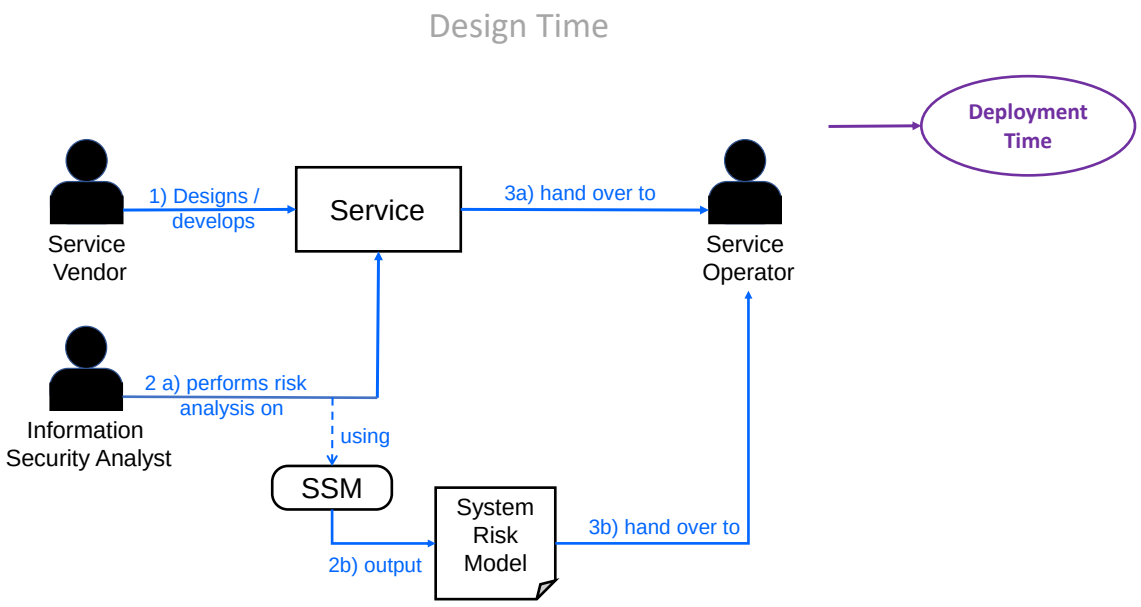

Figure 2.6: Part 1 of methodology for an interacting usage of SSM and CSAP tool

The second benefit is that an SSM model provides a machine understandable way to communicate this information. This is unambiguous and comprehensive, and also easy to extend by adding context information from the service operator's own analysis. This is done by using the CSAP tool to identify the information that should be added, and using SSM to integrate it with the supplier's model, as shown in Figure 2.7 and described in detail in Section .

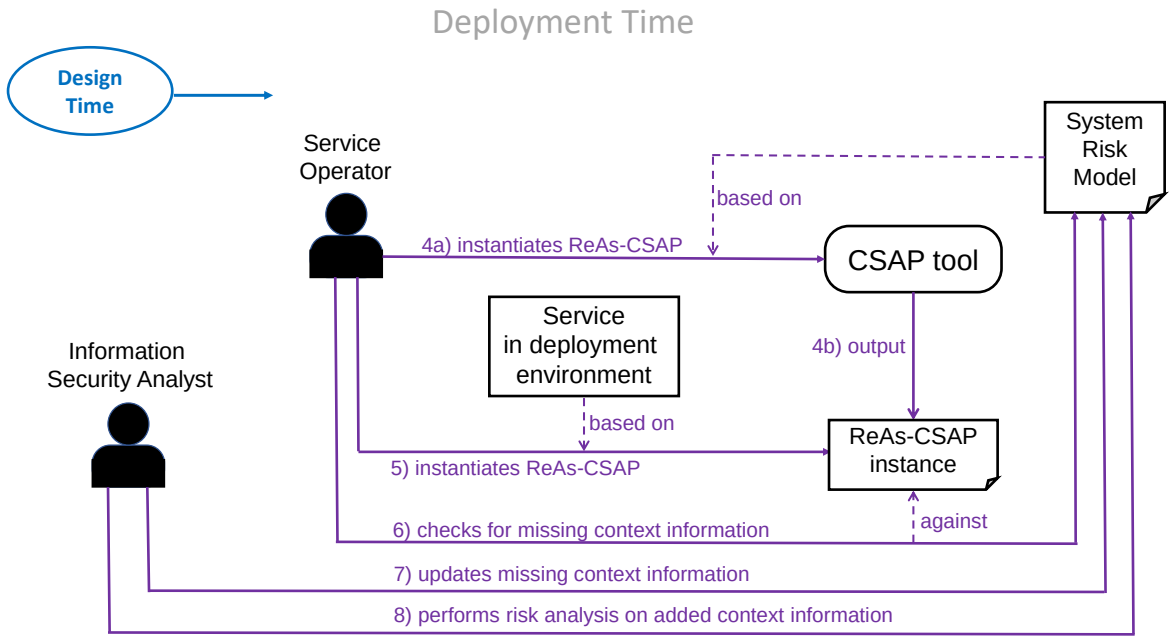

Figure 2.7: Part 2 of methodology for an interacting usage of SSM and CSAP tool

We envisage that in a typical scenario, the software/service supplier will produce an SSM model describing how their components interact with each other and with cloud resources, including assumptions on the deployment such as which services are assumed to be colocated, the impact of potential security breaches, what security measures are assumed to be in place including those provided by the supplied software/services. These last points may be supported by proof in the form of independent penetration tests, showing that the security measures included in the software/services can prevent threats they are supposed to address, as described in the SSM model. The conduct of these penetration tests is described in Section 2.2.2.

The service operator's own analysis will refine this information, adding security measures or deployment constraints, revising SSM model inputs describing the trustworthiness of third parties (including the cloud operator's staff and resources), and the business impact should primary assets be compromised. The service operator will also introduce assumptions about the jurisdictions where the application is deployed, and where its users are located. Once this is included, the resulting SSM model can be used to carry out the deployment phase risk assessment, and if necessary, further penetration tests to verify resistance to threats in the envisaged deployment environment. These stages are described in Sections 2.2.3 and 2.2.4.

At the end of this process, the SSM model provides a machine understandable way to transfer the risk assessment to the run-time phase, in conjunction with the RestAssured Runtime Risk Evaluator (RRE) tool, which is discussed separately in Section .

2.2.1 Design time risk assessment by the software/service supplier

At this stage the supplier (assumed also to be the developer) of the software/services to be used will carry out their own risk assessment. This should normally be done during the design of the software, and may be updated periodically if there are any changes to:

- the software itself
- the classes of threats to be considered, e.g. when a new type of attack is discovered; or
- the expected business or operating environment and associated context assumptions.

The first step is to determine the context and risk assessment procedures and criteria. RestAssured risk analysis tools support a specific approach based on ISO 27005. The scales for threat likelihood, impact and risk levels are specified in the knowledge base used with these tools (see Section 3.3.2. Each knowledge base also contains models of the types of assets from which systems are composed, and models of threats to those assets including causes, effects and control strategies. In practice, a software/service supplier using the RestAssured approach would need to choose a knowledge base that covered any specialised asset types they expect may be involved, e.g. specialised cloud platform features or types of networks, and edge devices such as mobile and IoT devices.

Subsequently, the software/service supplier should carry out the steps shown in Table 2.2.

Table 2.2: Risk analysis during software/service design and development

Task	Description	User	Software Life-Cycle	RestAssured Tool Support
1	Definition of the scope/boundaries for the risk assessment. Normally this covers a full system that uses the supplied software/services, based on assumptions about the deployment (cloud security measures) and the service operator's concerns.	Developer/ Software Architect	Design	This is a business decision, the results of which become part of the SSM model from tasks 3(a) and 3(b) - see later in this table.
2	Identification of relevant laws, regulations and other obligations of the service operator. Here, the software or service supplier may make reasonable assumptions or impose reasonable constraints.	Software Architect	Design	CSAP may be used if the assumptions or constraints are complex. Results are included in the SSM model from tasks 3(a) and 3(b).
3(a)	Identification of primary assets, i.e. information/data that is processed and stored by the software (e.g. user data of volunteers), and key business processes which are mediated by software.	Developer/ Software Architect	Design	Modelling primary system assets and their relationships using SSM, given suitable assumptions about the deployment.
3(b)	Identification of supporting assets, e.g. networks, devices (including virtual devices), physical spaces, and non-critical processes within the system. These normally include cloud resources which may be chosen by service operators, for which the software/service supplier will need to make reasonable assumptions or set reasonable constraints.	Developer/ Software Architect	Design	
4(a)	Assessment and modelling of impact if primary assets are compromised.	Customer Account Manager	Design	Modelling impact levels in SSM.

Table 2.2: Risk analysis during software/service design and development (cont.)

Task	Description	User	Software Life-Cycle	RestAssured Tool Support
4(b)	Assessment and modelling of impact if secondary assets are compromised.	Developer/ Software Architect	Design	SSM models asset dependencies, so it is not necessary to separately specify impact levels for secondary asset compromises.
5	Identification of threats	Developer/ Software Architect	Design	Automatic threat catalogue creation using SSM.
6	Identification of vulnerabilities, that are derived from unmitigated threats.	Developer/ Software Architect	Design	Auto detection of threats for which no mitigation has been specified.
7	Extraction of security objectives (SO) that should be tested, and the corresponding security functional requirements (SFR) and assurance requirements (SAR) for the involved assets.	Developer/ Software Architect	Design	Extraction using SSM of security measures and the threats addressed by them.
8	Identification of test scenarios	Developer/ Software Architect/ Penetration Tester	Design	
9	Specification of security measures that should be used to achieve the security objectives.	Developer/ Software Architect	Design	SSM provides hints on security controls that could be used to reduce threat likelihood.
10	Assessment of risk levels based on the likelihood of threats producing an effect and its impact, given the envisaged security measures.	Developer/ Software Architect	Design	Automatic risk level evaluation in SSM.
11	Risk treatment if risk level is not acceptable (e.g. adding additional or stronger measures for SF).	Developer/ Software Architect	Design	

The main RestAssured tool used by software/service suppliers is SSM, since this captures a high-level system design in terms of interacting processes, the devices where they are executed and where data is stored, the networks over which these devices communicate and physical locations where they can be accessed.

To create a model of the entire system, the software/service supplier must make assumptions on the service operator's concerns and deployment environment. This includes assumptions about the type of cloud that will be used, and the security measures implemented by the cloud resource provider. This is often easier for the software/service supplier to determine, since they understand in detail where their software depends on cloud platform features, and how those features are used to for security purposes such as user authentication.

Figure 2.8 shows the specific expression of the general risk assessment approach from Section 2.1.3.1 (see Figure 2.5). The steps in this specific risk assessment approach address the tasks provided in Table 2.2.

In the following, the differences between the design-time specific and the general approach are discussed. In Figure 2.8, new and changed tasks and artifacts are highlighted in purple.

During design-time, the context establishment is performed in two steps (see Figure 2.8, RestAssurred approach steps (1) and (2)). To emphasize that the created system model represents the software/service at design time, it is called the *design-time system model (DTMS)* (see Figure 2.8, RestAssurred approach step (3a) ff.). The extraction of security functional requirements (see Figure 2.8, RestAssurred approach step (8)) is a new task. It extracts the security functional requirements for the software/service that shall be implemented for the achievement of the corresponding security objectives. The security assurance requirements are extracted, in terms of the types of threats (from RestAssured step (5)) that should be addressed.

Based on the extracted security functional requirements, scenarios for the later testing of the software/service (see Section 2.2.2) are defined during its development (see Figure 2.8, RestAssurred approach step (9)). Because no controls are existing at the beginning of the design of the software/service, the identification of controls is performed at a later point (see Figure 2.8, RestAssured approach step (11)). Here, the identified controls define how the security functional requirements should be implemented, including how this depends on security measures included within the cloud computing services/infrastructures for the deployment of a software/service. These requirements address the achievement of appropriate security goals by the cloud computing service/infrastructure of a cloud service provider.

The final steps (12) and (13) from Figure 2.8 involve an automated calculation of residual risk levels using SSM, after which one can determine the appropriate risk treatment. If this involves further risk reduction measures, one returns to step (11) to include more security measures in the SSM model, or (if that is not possible) steps (1-3) to change the system assets or composition.

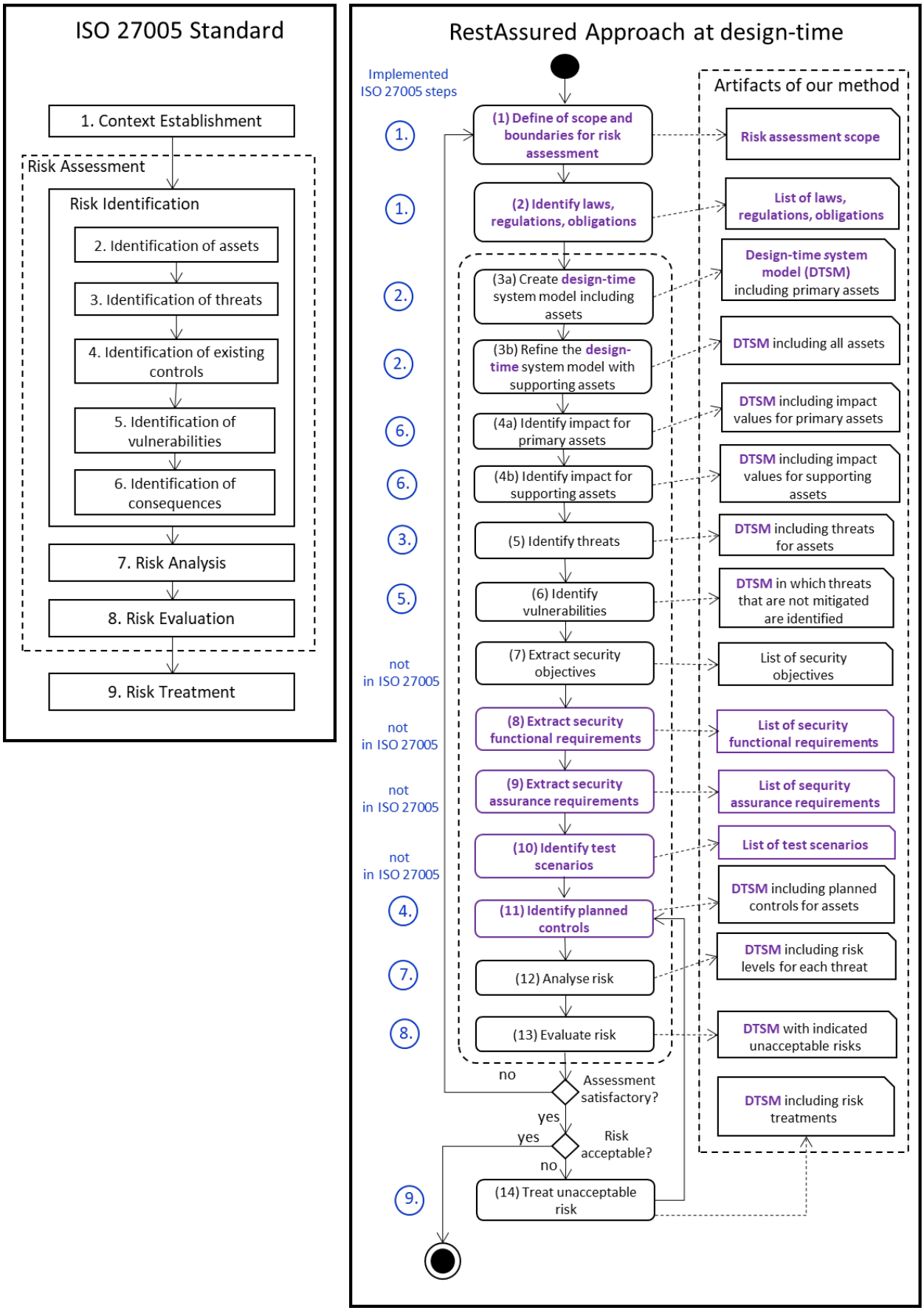

Figure 2.8: RestAssured approach for risk assessment during design-time compared to ISO 27005

2.2.2 Independent penetration testing of software/services

The procedure for testing the security of software and services consists of the steps from Table 2.3:

Table 2.3: Revaluation of the software/service implementation

Task	Description	User	Software Life-Cycle	RestAssured Too Support
1	Evaluation of the design of the software/services regarding possible vulnerabilities.	Security Analyst/ Penetration Testert	Security Evaluation	None
2(a)	Check that the specified Security Functionality has been implemented by e.g. performing code reviews.	Security Analyst/ Penetration Tester	Security Evaluation	None
2(b)	Performing penetration tests on exposed service endpoints and APIs to confirm that: • the strength of security controls is sufficient, e.g. encryption algorithms and key lengths • potential vulnerabilities cannot be exploited, e.g. using injection attacks, cross-site scripting attacks, etc.	Penetration Tester	Security Evaluation	None, other than from SSM models describing the SO, SRF and SAR.
2(c)	Performing penetration tests on supporting assets (e.g. device operating systems, etc).	Penetration Tester	Security Evaluation	
3	Examine the security of the development environment used to create software/ services.	Security Analyst	Security Evaluation	None

The security evaluation process should provide evidence for the security properties of the software or service components provided by their supplier. If the tests are made by an independent 3rd party, this provides additional credibility for the security claims.

Results from the tests, and especially from steps 2(a) and 2(b) above, should be fed back into the design-time risk model produced using SSM. This model will then provide a rigorous specification of the security and context assumptions used for risk assessment and a justification for which security objectives and requirements were tested.

From these results, the software/service supplier may wish to produce a specification or guidelines for the deployment of the software/services in an operational context. This will include security controls for which the service operator must be responsible, e.g. the secure configuration of devices, use of strong passwords, etc. It may also describe the assumptions made about the deployment environment, e.g. on the choice of cloud resource provider, or the use of cloud platform facilities.

2.2.3 Deployment phase risk assessment by the service operator

The final risk assessment is then made by the service operator when considering a specific deployment of a system containing the supplied software/services. Results from the design time risk analysis (Section 2.2.1) and security assessment (Section 2.2.2) are used, incorporated into an initial SSM model. This model is then expanded and made more specific by adding further supporting assets and context in the service operator's intended deployment environment.

The steps required here are described in Table 2.4:

Table 2.4: Risk analysis at deployment time

Task	Description	User	Software Life-Cycle	RestAssured Too Support
1	Definition of the scope/boundaries for the risk assessment. This will be the system about to be deployed, and the deployment environment.	Security Analyst	Deployment	CSAP analysis using RestAssured CSAP patterns.
2	Addition of supporting deployment assets and, where relevant, additional stakeholders and any jurisdictional or regulatory constraints.	Security Analyst	Deployment	SSM, using outputs from a CSAP model created in Step 1.
3	Assessment of impact for supporting deployment assets that are identified during deployment	Security Analyst	Deployment	Modelling impact levels in SSM, if not already covered by SSM dependencies analysis.
4	Identification of threats for supporting deployment assets that are identified during deployment	Security Analyst	Deployment	Automatic threat catalogue creation using SSM.
5	Identification of vulnerabilities for supporting deployment assets that are identified during deployment. In this connection, a vulnerability is a threat that is not mitigated by a control	Security Analyst	Deployment	Auto detection of threats for which no mitigation has been specified.
6	Extraction of security objectives (SO) and security functional requirements (SFR) for the envisaged deployment.	Security Analyst	Deployment	SSM reporting functionality.
7	Specification of security measures that should be used to achieve the security objectives.	Security Analyst	Deployment	SSM provides hints on security controls that could be used to reduce threat likelihood.
8	Assessment of risk levels taking account of new assets, vulnerabilities and controls.	Security Analyst	Deployment	Automatic risk level evaluation in SSM.
9	Risk treatment if risk level is not acceptable (e.g. adding additional or stronger SARs).	Security Analyst	Deployment	

The result of this risk assessment determines whether the envisaged deployment will be safe, given the service operator's business requirements and any regulatory compliance requirements affecting them.

Figure 2.9 shows the specific expression of the general risk assessment approach from Section 2.1.3.1 (see Figure 2.5). The steps in this specific risk assessment approach address the tasks provided in Table 2.4. In the following, the differences between the deployment specific and the general approach are discussed. In Figure 2.9, new and changed tasks and artifacts are highlighted in purple.

For the context establishment regarding the deployment environment of the software/service a ReAs-CSAP instance is created (see Figure 2.9, RestAssured approach step (1)). In this connection, the appropriate context information from the design-time risk assessment (see Table 2.2) shall be considered.

The design-time system model (see 2.8), which has been created during the design-time risk assessment, represents the basis for the risk assessment during the deployment. This system model is expanded by supporting assets of the deployment environment that are named *supporting deployment assets* (see Figure 2.9, RestAssured approach step (2)). For the added supporting deployment assets the impact, threats and vulnerabilities are identified (see Figure 2.9, RestAssured approach steps (3), (4) and (5)). In new steps, the security functional requirements for the deployment environment that shall achieve the security objectives are extracted, and security controls identified (see Figure 2.9, RestAssured approach steps (6), (7), (8) and (9)). In steps (10) and (11), the risk levels are determined automatically by SSM, and an evaluation made to determine the appropriate risk treatment (see also Figure 2.9. Step (11) may require further reduction in risk levels, in which case steps (9)-(11) are repeated. Once the risk levels are acceptable, the controls should be implemented from step (9) to protect both primary assets and the supporting deployment assets.

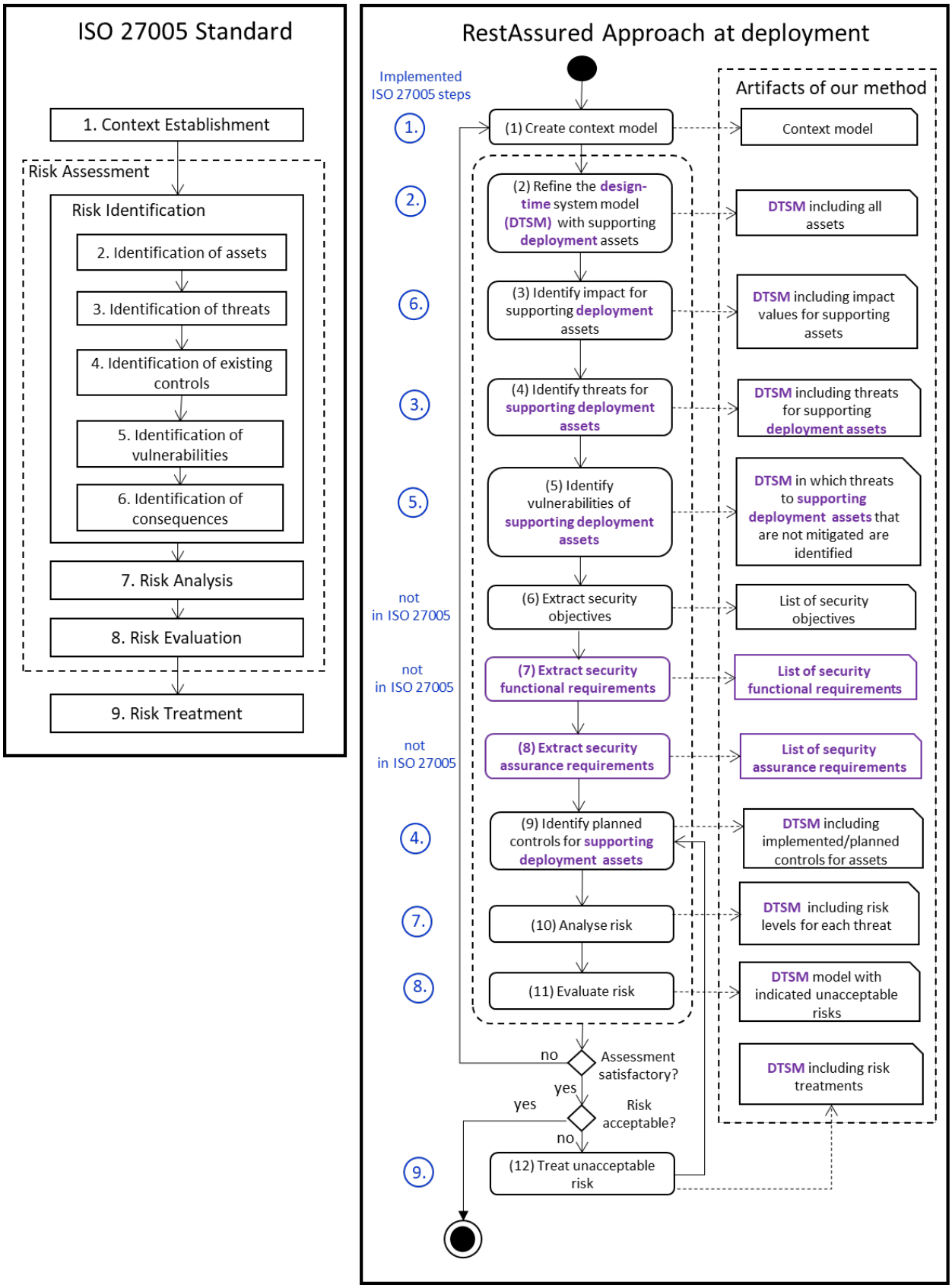

Figure 2.9: RestAssured approach for risk assessment during deployment compared to ISO 27005

2.2.4 Penetration testing by the service operator

In some cases, the service operator may wish to conduct additional penetration tests to verify security in the specific operational deployment environment. This may be necessary where the software/service supplier cannot arrange penetration testing in a sufficiently realistic environment. This may arise if:

- the service operator wishes to deploy software/services in an environment that is unlike that envisaged by the software/service supplier; or
- the deployment environment is only accessible to the service operator, e.g. because it uses a hybrid cloud composed partly of in-house resources.

In these cases, if the service operator needs to verify security properties, they should carry out their own tests, where permitted by the cloud (or other) resource providers.

2.3 Risk Levels and Adaptation

In a modern, cloud-based system composed of micro-services, the configuration of the system is not static. The above tools can capture the intended structure of the system (which types of services are connected together in which ways), but not the configuration (which services, deployed where, with what security measures, and what dataflows). This can only be determined when the system is deployed, creating specific service networks hosted at specific locations under known jurisdictions. The configuration may then change due to performance and cost optimisation adaptations, which will most likely be triggered by an autonomic cloud infrastructure management system. Changes may also become necessary due to failures in certain system components, forcing the system to adapt to use alternatives.

We envisage that the machine understandable models produced at design time will also be used at run-time to evaluate risks during the operation of the system. The idea is that the design time model specifies the intended structure of the system as defined by its designer, including security measures that could be used to protect system assets. For each type of threat identified at design time, we can extract the assets involved, and find the control strategies (combinations of security measures) that are considered adequate to manage the risk.

Figure 2.10 provides a conceptual view of the logical components and interfaces required for risk assessment and adaptation in RestAssured.

The direct monitoring of the system (including application and platform or infrastructure components) is handled by the System Monitor component in Figure 2.10. This is part of adaptation system from WP5, whose job is to create and maintain the run-time system model containing information about components present in the system and their relationships. The System Analyser component detects when changes affect the objectives for autonomic adaptation, which are in general not limited to security and data protection objectives. It then triggers the Adaptation Planner to generate an adaptation plan, which can be executed by the Adaptation Engine to maintain compliance with those objectives. As part of this overall adaptation procedure, the System Analyser and Adaptation Planner components interact with the Runtime Risk Evaluator component from WP7 to obtain information about security risk levels.

If security and data protection were not a concern, typical adaptation objectives might be to maximise the performance of an application subject to a cost constraint, or to minimise costs while maintaining some minimum level of performance. Once security and data protection are included in this picture, two extra clauses must be added to the adaptation objectives:

- the level of risks to data in the system must not be allowed to exceed some upper limit;
- the use of a risk assessment during the adaptation decision process must be recorded.

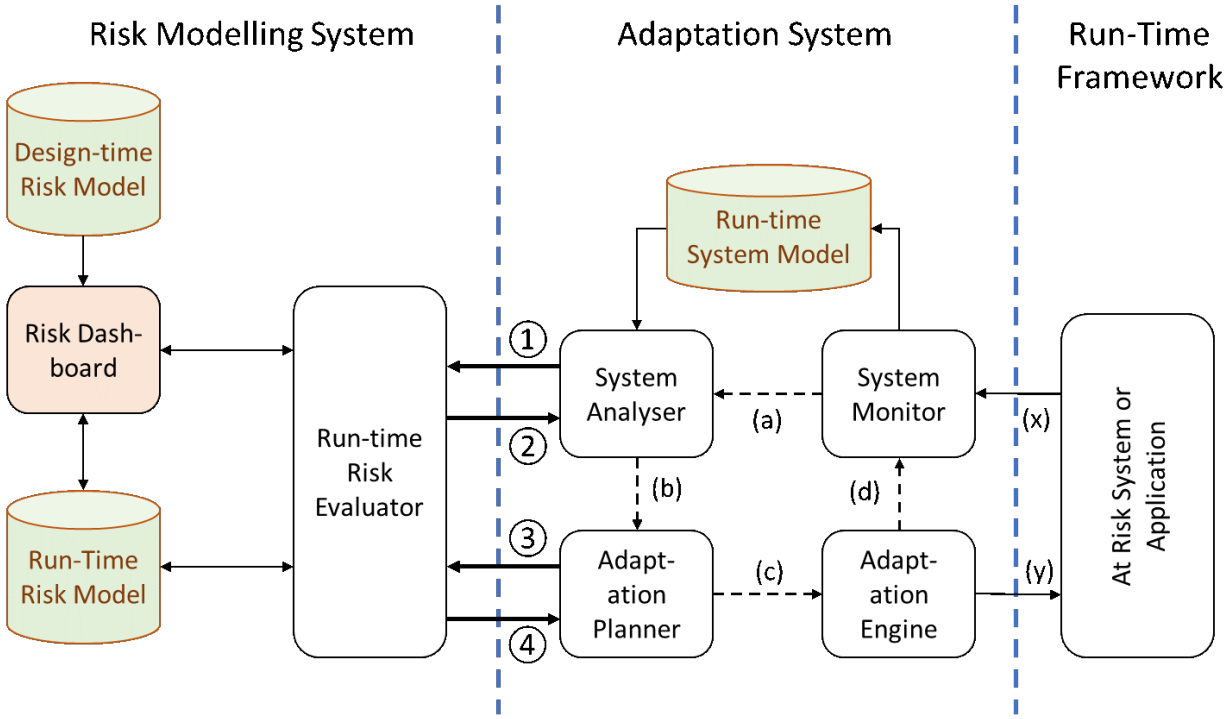

Figure 2.10: Run-time risk assessment in RestAssured

The first of these extra clauses ensures that data is protected sufficiently well given the threats to which it may be exposed, required by the GDPR 'at the time of processing' (i.e. at run-time). What it means in practice is that a risk assessment must be made whenever a change to the system is being considered, and whenever the system changes (for any reason not limited to a planned adaptation). The second clause is necessary because the GDPR requires data controllers and processes to show that the security measures used are appropriate to the risks, so an auditable record showing that risks were assessed is needed. This is of course very difficult to achieve in an autonomically managed system or infrastructure unless one has access to RestAssured solutions.

To manage the level of risks, the adaptation system must obtain an updated risk evaluation if the system changes in ways that may affect the security of protected data. The System Analyser can decide which types of changes meet this criterion, bearing in mind that (since this is a risk evaluation) if in doubt it should request an update. It then sends a change notification message on channel ① in Figure 2.10 to the Runtime Risk Evaluator component from WP7. This sends back a report via channel ② giving the new risk level in the system and the threats responsible for it. The System Analyser can then decide whether the new risk levels are sufficiently close to the upper limit that action should be taken to reduce the risks. If they are, it signals the Adaptation Planner to find a suitable adaptation that would have that effect.

It is also possible that the adaptation system may wish to trigger a change for other reasons, e.g. to optimise the use of cloud resources, improve the balance between cost and performance, or to respond to a software crash or hardware failure. Before it makes any changes, it should signal the risk evaluator using channel ③ in Figure 2.10. When the risk evaluator receives such a change proposal, it will evaluate and report the risk level to the adaptation component, using channel ④ in Figure 2.10. The Adaptation Planner then decides whether to go ahead with the planned change (by passing it to the Adaptation Engine), try a different plan, or leave the system as it is.

The information supplied to the Runtime Risk Evaluator on channels ① and ③ should include a list of assets that have (or would be) added or removed, plus the relationships of added assets to those already in the system, and the security measures available at the added assets. Migration of a software component from one hardware node to another could therefore be represented by removing one asset and adding another.

In addition, the RRE needs information about the security measures implemented for each new asset, or changes in security for existing assets. At this stage, the System Analyser adaptation component passes the complete run-time model as a json object, and the Runtime Risk Evaluator decides what has changed, and which changes should be made in the run-time risk model in Figure 2.10.

Information about the behaviour and status of assets would also be useful, as this may allow changes in the trustworthiness of certain assets to be detected. However, monitoring asset trustworthiness is beyond the scope of RestAssured, and even some security controls may be impossible to monitor directly. At this stage we envisage that changes in trustworthiness assumptions will be provided as user input via the Risk Dashboard component in Figure 2.10, along with organisational security measures that cannot be detected in the run-time system. The risk dashboard will also provide a simple display showing risks in the system.

The Runtime Risk Evaluator will use information derived from the design-time risk model, obtained from SSM, as indicated in Figure 2.10. This is discussed in more detail in Section 4.3.

The response from the Run-time Risk Evaluator on channels ② and ④ gives the worst-case risk level in the system including the communicated changes, the asset compromises for which this risk level applies (there may be more than one), and the list of threats assessed as being likely to cause these compromises.

The round trip via ① and ② or via ③ and ④ does need to be executed in the run-time management loop. It is possible that sending the full run-time system model contents will take too long, in which case it may be necessary to move some of the change detection logic to the System Analyser, so it can send only relevant changes to the RRE. At this stage, the risk level update calculation is expected to take the most time, and it isn't yet totally clear which changes will affect the risk levels, so optimising the volume of data exchanged has been left for a later release.

It has also become clear that the mapping between the run-time system model and the risk model is more complex than it first appeared. This is because risks can only be established in the context of a business purpose, so the risk model is necessarily defined in that context and uses trustworthiness criteria to express the effect of interactions between (non-dedicated) assets and other business actors and purposes. However, system monitoring functions cannot determine the business purpose of a component, so the run-time system model has a wider scope with no mechanism for describing external influences. At this stage, the focus is to develop the necessary mappings by starting from the run-time system model, and this is one of the reasons why a separate risk monitoring process has been dropped from Figure 2.10. It may still be possible to manually trigger an adaptation if new information is supplied by a user via the Risk Dashboard. This will be considered in D7.3.

2.4 Summary

The overall security and privacy engineering methodology is based on the analysis and treatment of risks, both at design time and during run-time. The benefits of this methodology are:

- it is well aligned to existing information security standards including ISO 27005 and ISO 27001, avoiding possible problems with adoption by standards compliant organisations;
- it allows users to supply all the key pieces of input, including trustworthiness and impact assumptions, while automating complex calculations of risk;
- it supports run-time as well as design-time risk assessment and management, making it possible to fulfil the General Data Protection Regulation (GDPR) [7] requirement for continuous risk assessment for any changes.

Since the start of the second project period, some refinements have been made to the overall procedure, in two main areas.

The interaction between CSAP and SSM has been reassessed in light of input from OCC and Adaptant on the need to capture information at each level at the appropriate time and from the appropriate stakeholder.

Initially we assumed CSAP would be used first to capture context, and SSM would then be used to fill in the architectural details for a specific system or application. We now recognise that the architecture is defined by the supplier of software and service components, and only later does the system or application operator decide in what context to deploy these components. Thus the workflow presented in Section 2.2 is different and more complex than the one described in Deliverable D7.1 [43].

The other significant change in the overall methodology is in the interaction between WP5 adaptation components and WP7 risk evaluation components. The initial design described in D7.1 has been analysed in more detail, and the overall approach has been simplified by removing the separate risk monitor component. This is partly in recognition of the fact that mapping between the run-time system model from WP5 and the risk models in WP7 is more complex than it first appeared, and it makes sense to focus on mapping in one direction at a time. However, it may be possible to manually initiate an adaptation if new information is supplied by a user via the Risk Dashboard. This will be considered in D7.3.

3 Models

As already described in Section 2.2, in RestAssured we have two risk analysis tools, CSAP and System Security Modeller, that model different aspects of a system. CSAP models the overall context for the system, while SSM models the detailed relationships between the assets of the system. In this chapter we describe the underpinning threat models of these two tools.

3.1 Modelling the System Context

In this section[1], we explain how to perform a context analysis using the RestAssured-Cloud System Analysis Pattern (ReAs-CSAP).

Section 3.1.1 first presents problems in defining the context of a cloud system and then we present our ReAs-CSAP to address those problems. The different elements of the ReAs-CSAP are discussed in Section 3.1.2. The instantiation of the ReAs-CSAP is already presented in D7.1 using CSAP tool. Section 3.1.2 discusses the result of the state of the art analysis.

3.1.1 Problems

As mentioned earlier, a definition of the context is needed for performing a risk assessment for a system. This context definition has to represent information about the external and the internal context of a cloud computing service. With respect to the external context, it is necessary to identify all the requirements that are indirectly relevant for the provided service and the corresponding system. Here, information related to legal, regulatory and contractual requirements has to be considered. The internal context has to consider the particular cloud computing service and its supporting system itself. The following information has to be specified:

- relevant legal, regulatory and contractual requirements
- relevant internal stakeholders that are interacting with the provided service(s)
- relevant locations
- high-level primary assets; such assets represent business processes of an organization and information that is related to the business processes.
- high-level supporting assets; such assets support the execution of primary assets in the business processes, by processing, storing and transmitting the related information (e.g. storage, software, network components).
- interfaces (cf. ISO/IEC 27005 [22], p. 12)

For getting meaningful results from the risk analysis, it is necessary that the above-mentioned information regarding the external and internal system context is complete. It has to be ensured that all different types of context information are considered with respect to its relevance. If the context definition is performed without any further support, there is a risk that particular types of context information are not considered. Furthermore, the relations between different pieces of context information have to be defined, because during the assessment of particular risks, the corresponding interactions have to be considered.

Our solution is a pattern-based approach for analyzing the context of a cloud computing service. In Section 3.1.2, we present the solution part in more detail.

[1]Note that part of this section has been published in Future Internet Journal (an open access journal) https://www.mdpi.com/journal/futureinternet (accessed on 5 September 2019), see [13].

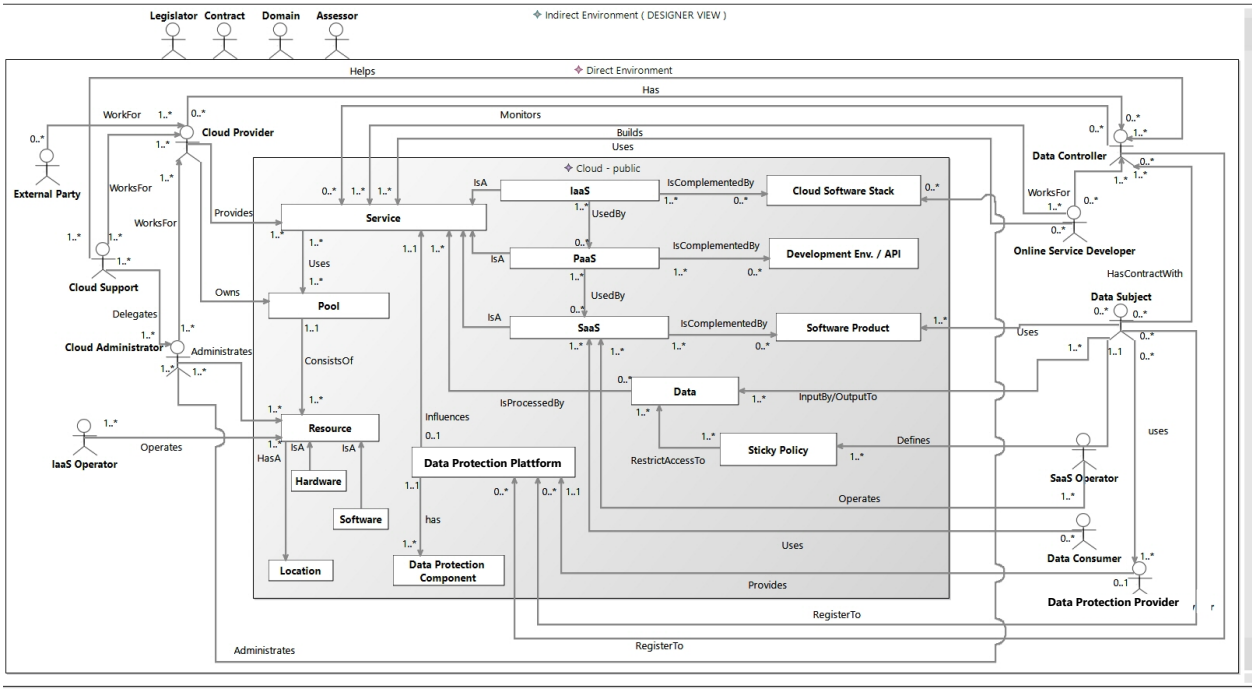

Figure 3.1: RestAssured-Cloud System Analysis Pattern (ReAs-CSAP)

3.1.2 Description of the ReAs-CSAP

The ReAs-CSAP provides model elements for defining the different types of context information and their relations to cloud computing services. Thus, it can be ensured that neither types of context information nor the relevant relations between particular types of context information are overlooked unintentionally during the context definition.

Because its model elements are defined by a meta-model, the ReAs-CSAP enables a terminology-based context definition. Additionally, the ReAs-CSAP provides a graphical representation of the corresponding model elements. Thus, a good overview of the specific context information is given and an intuitive use of the ReAs-CSAP is made possible. A tool support for the definition of the ReAs-CSAP and its instantiation is provided by the *ClouDAT-tool* (see Section 4.1).

This subsection contains a detailed description of the elements provided by the ReAs-CSAP. The ReAs-CSAP expands the CSAP from [4] with new types of context information that allows a representation of 1) particular information that is relevant for data protection, and 2) information regarding the compliance to laws and regulations for security and privacy information.

The ReAs-CSAP is shown in Figure 3.1. In the following, we discuss the different elements of the ReAs-CSAP. Here, the ReAs-CSAP elements are associated with the problems of a context definition from Section 3.1.1. The information of the internal context (see Section 3.1.1) is represented by the *Direct Environment*. The *Direct Environment* contains the *Direct Stakeholders* that are relevant for the considered cloud computing services and the *Cloud* itself. A Cloud is represented graphically by a grey box (see Figure 3.1). The Cloud represents its different parts in the form of *Cloud Elements*.

Cloud Elements specify the particular cloud computing services and the physical resources of the Cloud. Graphically, the Cloud Elements are represented by white boxes inside the Cloud (see Figure 3.1). The associations between the different Cloud Elements are also represented. Direct Stakeholders are able to interact with Cloud Elements. The logical relationships between Direct Stakeholders are also considered within the ReAs-CSAP. The different types of direct Stakeholders are the following:

Data Subject is an identified or identifiable natural person who uses the considered cloud computing ser-

vice. In this context, personally identifiable information and/or sensitive personal information (SPI) of the Data Subject is processed and/or stored by using the cloud computing service. Data Subject represents an important role, because assuring the privacy and security of their data is a main goal.

Data Controller is a provider of the considered cloud computing service in a form of Infrastructure as a Service (IaaS), Platform as a Service (PaaS) or Software as a Service (SaaS). Data controllers have contracts with the Data Subjects who use the provided cloud computing services. A Data Controller is legally responsible for the compliance of the specified privacy and security requirements for the Data Subjects' data. For ensuring this compliance, Data Controllers make use of data protection concepts and technologies. For making use of such technologies, Data Controllers can register to the Data Protection Platform. Depending on the level of the provided cloud computing service, Data Controller can use either their own IaaS and/or PaaS or an external IaaS and/or PaaS infrastructure provided by particular Cloud Providers.

Data Consumers are natural or legal persons, public authorities, agencies or any other bodies. Besides the Data Subjects, they are also users of the considered cloud computing service. However, their use case can be different from the use case of the Data Subjects. Within this use case the accessible data of Data Subjects is disclosed to Data Consumers. The rights for Data Consumers with respect to the access to and usage of specific data is defined in the corresponding Sticky Policies.

Cloud Providers are legal entities that provide cloud computing services in the form of IaaS and/or PaaS that are used for providing a SaaS. In the case of *IaaS Providers*, they also own the resources for providing this type of cloud computing service. Cloud Providers can have associations to the other types of direct stakeholders who are working for Cloud Providers as follows:

- The optional *Cloud Support* works for the Cloud Provider. It represents the point of contact for Cloud Customers if they have questions or problems related to the used IaaS and/or PaaS cloud computing services. Possible problems are delegated to the Cloud Administrators.
- *Cloud Administrators* work for Cloud Providers. They administrate the Resources of the Cloud as well as the Cloud Software Stack and handle problems that have been reported by customers.
- *External Parties* specify service providers that work for the Cloud Provider. Here, an External Party delivers services that are relevant for the cloud or affect the operation of the considered cloud computing service. For example, external parties could be represented by companies for the maintenance of IT-resources and air-condition or cleaning services.
- *IaaS Operators* perform tasks for the operation of an IaaS.

Online Service Developer specifies a legal person or organization that has developed the Software Product that is provided by the Data Controller via the corresponding SaaS. Online Service Developers work for the Data Controller. They need profound skills of the technologies (e.g. Intel Software Guard Extensions (SGX) [16, 2]) that are used by the Data Protection Platform.

External Data Protection Service Providers host the components of the Data Protection Platform in an according infrastructure and provide the functionality of these components as a service. This service can be used by Data Controllers that do not want to be in charge of hosting the Data Protection Components.

SaaS Operators perform tasks for the operation of a SaaS.

Beside the Direct Stakeholders, the Direct Environment also contains the *Cloud* that in turn contains the different *Cloud Elements*. The Cloud can represent a public, private, hybrid or community cloud. In the graphical representation of the ReAs-CSAP the type of the Cloud is displayed in the Cloud beside the key word "Cloud" (see Figure 3.1). The contained Cloud Elements are as follows:

- High-level primary assets in the form of business processes specified by (cloud computing) *Services* (IaaS, PaaS, SaaS)
- High-level primary assets in the form of information specified by *Data*
- High-level supporting assets specified by *Resources* (Hardware, Software, Location)
- Data protection concepts and technologies (Sticky Policies, Data Protection Platform)

The different types of Cloud Elements are the following:

Service defines a central point for referencing all provided cloud computing services. Data has an association with Service, because the relevant Data is processed by all provided cloud computing services. The *Data Protection Platform* influences the processing of the Data on the PaaS level.

SaaS represents a cloud computing service on the SaaS level. SaaS is complemented by the Software Product(s) whose functionality is provided via the SaaS. SaaS uses a PaaS for the provision of its service.

PaaS specifies a cloud computing service on the PaaS level that is used by the SaaS. PaaS is complemented by the Development Environment and Application Programming Interface (API) that is provided via the PaaS. PaaS uses the resources that are provided by an IaaS. A PaaS is provided by a Cloud Provider.

IaaS defines a cloud computing service on the IaaS level that provides the necessary infrastructure with different types of resources. Examples for such *Resources* are hardware for storage and processing power. Resources are used directly by the PaaS and indirectly by the SaaS. IaaS is complemented by the Cloud Software Stack and the infrastructure resources like Software and Hardware. The IaaS and the Cloud Software Stack are administrated by the Cloud Administrator.

Cloud Software Stack represents the cloud software stack that is necessary for providing the corresponding IaaS.

Development Environment and API specifies the API and Development Environment that is provided by the according PaaS. The provided API and Development Environment are used for developing the Software Product that is provided by the considered SaaS. The API provides functionalities that enable the use of the relevant PaaS resources and IaaS resources by the Software Product without the need of knowing any specific technical details concerning these resources.

Software Product represents the software that is provided by the Data Controller via the corresponding SaaS. The Software Product has been developed by the Online Service Developer. For the development, the API and Development Environment of the PaaS are used.

Pool is the central point for referencing all relevant physical resources of the Cloud that are necessary for providing the appropriate cloud computing services.

Resource is the central point for referencing all resources in the form of Locations, Software and Hardware.

Location represents all locations that contain cloud resources (e.g. computing center) or are relevant for the provided cloud computing service in another way (e.g. development site).

Hardware represents necessary cloud hardware resources, e.g. server racks or network components.

Software represents cloud resources in the form of necessary software (e.g. software for managing the cloud or virtualization).

Data specifies the personally identifiable information and/or sensitive personal information of the Data Subject that is processed and/or stored by the according SaaS.

The rights for accessing and using this data by Data Consumers are specified in 1) a legal specification by the contract between the Data Subject and the Data Controller 2) a formal specification in the associated Sticky Policies.

Sticky Policy defines requirements regarding the access and usage of the Data of a Data Subject. They are derived from the Contract between the Data Subject and Data Controller.

Data Protection Platform provides security and privacy mechanisms. These mechanisms constrain the PaaS level for enforcing the relevant Sticky Policies (e.g. in the transfer, processing and storage). The Data Protection Platform might provide different Data Protection Components that implement the appropriate security and privacy mechanisms. The different components can be hosted by either *Data Controllers* themselves in an according infrastructure or provided by *Data Protection Providers* that provide the functionality of the components as a service.

Data Protection Component defines the components that implement security and privacy mechanisms for the considered cloud computing service. *Data Protection Component* could include concrete technologies for data protection, such as attribute-based encryption (e.g. [44]), fully homomorphic encryption (e.g. [26]) or secure enclaves (e.g. [16, 2]).

The *Indirect Environment* represents the external context of a cloud computing service. The relevant information about the external context is represented by different types of *Indirect Stakeholders* that are contained in the *Indirect Environment*. Here, the different types of *Indirect Stakeholders* represent the following information:

Legislator: Representation of laws and regulations of legislators (e.g. Germany or the European Union) that are relevant for the cloud computing service. This type of Indirect Stakeholder is especially important, because it enables the representation of laws regarding data privacy. In the context of the European Union, the *GDPR* is especially relevant. Relevant Legislators can be derived from the *Location* of the considered cloud computing service.

Domain: Specification of domain-specific rules and guidelines the cloud computing service has to comply with.

Contract: Representation of contractual provisions (e.g. Service Level Agreements with customers) that have to be fulfilled by the cloud computing service. The representation of the Contracts between Data Subjects and Data Controllers are particularly important because they specify the security and privacy requirements for the data of the Data Subjects. The Sticky Policies that specify these privacy requirements in a machine-readable way are derived from such Contracts. It should be ensured that the content of a Contract conforms to the relevant Legislators and Domains.

Assessor: Evaluating the level of security and/or privacy that is provided by the considered cloud computing service. The evaluation can be performed with respect to an existing standard such as ISO 27001 [20]. Assessors could analyze the results of the performed risk assessment. Here, they could map information of the risk assessment to the real system. For example, they could check the implementation of countermeasures that reduce the levels of corresponding risks.

3.1.2.1 Consequences

Our pattern allows the definition of the context for a cloud computing service. Because of the pattern approach, it is ensured that no relevant information is overlooked during the context definition. The definition

of the pattern syntax by a meta-model enables the analysis and further processing of the information provided by the pattern. This meta-model allows the extension of the pattern by new elements. Thus, the pattern can be adapted to address future needs.

The graphical representation of the pattern facilitates the context definition for users that are not so familiar with that subject. Furthermore, it supports identifiying the relationships between the different elements of the pattern.

By including elements that are relevant in the context of the GDPR, our pattern refers to the most important regulation for privacy and data protection in the European Union. To enhance the definition of the scope in conformance to ISO 27001, our pattern can be used in the context of an important widespread security standard.

3.1.3 The Result of the State of the Art Analysis

There are some works in requirement engineering that deal with security at the early stages of system design. For instance, Chung provides the non-functional requirement framework [6]. In his work, security is a class of non-functional requirements. For modelling security aspects in UML, Use Case models are extended, for example by misuse cases [1] and abuse cases [32]. Abuse frames [30, 29] extend Jackson's problem frames [24] to model security aspects. As a supplement to the above-mentioned works, our pattern complements them with a tighter integration of context analysis prior to the execution of risk assessment or requirements analysis methods at design-time. This may also guarantee a better conformance to standards as well as business and IT alignment.

Fredriksen et al. [12] propose the CORAS framework that supports a model-based risk management process. Prior to applying this process or any other risk assessment methods, our pattern can be applied for establishing the scope for risk management in the domain of cloud computing services.

Naudet et al. [36] propose a security requirements engineering process that consists of the following four steps: Context analysis and asset identification, security goal determination, refinement of these goals to security requirements, and countermeasures selection. Complementary to requirements engineering methods like [36], our pattern-based approach provides a systematic and structured execution of the context analysis. Our pattern can enhance the first step of the security requirements engineering process provided by Naudet et al. [36].

Haley et al. [15] propose a framework which unifies functional requirements from requirements engineering and assets as well as threats from security engineering. The focus of this work is the transformation of assets and threats into constraints of functional requirements. In the provided framework, the security requirements are identified based on the system context. The system context is specified using Jackson's problem diagrams [24]. In contrast to this usage of problem diagrams, our pattern uses a specific graphical notation for the definition of a system context that is oriented on the Unified Modeling Language (UML) [2].

Fenz et al. [8] present an ontology-based framework for preparing ISO 27001 audits. They provide a rule-based engine which uses a security ontology to determine if security requirements of a company are fulfilled. This work has no impact on ours, because we focus on different aspects of risk management.

Reusable patterns for security requirements have been studied by several researchers. For instance, Kis defines anti-patterns [25] for the identification of vulnerabilities and security requirements analysis at the business level. This work has no impact on our pattern, because our pattern does not represent an anti-pattern and has a different purpose.

With the focus on software systems, Sindre et al. propose a reuse-based approach to determine security requirements [41]. Firesmith also provides reusable templates for specifying security requirements [11]. In this context, [41] as well as [11] provide an asset-based risk-driven procedure for the identification of security requirements. This procedure contains the identification of valuable assets whereby no context for

[2] http://www.uml.org

the asset identification is defined. Our pattern supports the definition of a context that is a good basis for the asset identification. Hence, our pattern can be used complementarily to these methods.

Fernandez et al. [10] provide a unified way of representing all the components of a cloud ecosystem as well as security aspects. Our pattern supports the specification of the indirect and direct environment of a cloud system as well as its stakeholders, in addition to the cloud system itself.

In another work, Fernandez et al. [9] provide a method for building a security reference architecture for cloud systems. Furthermore, this paper presents a reference architecture for cloud computing environments using a class diagram. This class diagram represents the cloud services, the cloud and its components as well as cloud consumers and administrators. Our pattern considers the direct and indirect environment of a cloud system, in addition. Furthermore, all relevant direct stakeholders and their interactions with the cloud and among each other are represented. Specific types of direct stakeholders constitute roles that conform to the GDPR. Furthermore, our pattern can be used for the definition of the scope for a risk management with respect to the ISO 27005.

Additional to the security requirement patterns, there are several design patterns for security design aspects such as [27], [40], [28] for specifying security measures. A design pattern addresses a particular category of security concerns. The design patterns are generally used after the requirements engineering phase. In contrast, our work specifically focuses on identifying relevant context information and defining the scope for a risk assessment, especially during the requirements engineering phase. We focus on analyzing context and defining the scope for a risk analysis and assessment, hence the above-mentioned patterns can be used complementarily to ours.

Research on reusable patterns for context analysis has also received attention recently. Beckers et al. [5] provide a cloud system analysis pattern for establishing the context of cloud computing systems, expressed in a UML-like notation. In another work, Beckers et al. [4] present a method for eliciting security requirements for cloud computing systems. They define a terminology for the Cloud System Analysis Pattern (CSAP) from [5] with a meta-model. A concept for textual security requirement patterns that refer to the CSAP is also provided. We build our approach upon this work and present more specific pattern elements for addressing data protection goals in the context of cloud computing services. For this purpose, our pattern extends the CSAP from [4] by adding elements that represent roles as well as concepts and technical components regarding privacy concerns. The added roles conform to the GDPR. Furthermore, we provide information concerning logical relations of pattern elements that should be considered during the instantiation of the pattern.

In another paper, Beckers et al. [3] provide a catalog of security requirement patterns that are relevant for cloud computing. For ordering these security requirement patterns, different domains are defined. Furthermore, a structure for the representation of the security requirement patterns is represented. Because our pattern focuses on context definition, it could be used complementarily to this approach.

Lyubimov et al. [31] present a framework that supports eliciting the requirements for the realization of an ISMS conforming to ISO 27001. Their framework provides representations of different requirements from the ISO 27001 using models. One model considers the requirements with respect to the definition of the ISMS scope in an abstract way. As a specialization, our pattern focuses on the definition of a scope that is specific for cloud computing services. It supports the scope definition on a lower degree of abstraction, and provides concrete types of information that should be considered during the scope definition. Thus, our pattern could support the above-mentioned framework in the domain of cloud computing.

Montesino et al. [34] investigate the possible automation of controls that are listed in ISO 27001 and NIST SP800-53 [35] (catalog of security controls from National Institute of Standards and Technology (NIST)). This work considers the planning phase (from ISO 27001) of an ISMS. In contrast, our pattern focuses on the definition of a context. It can be used for defining the scope of an ISMS for a cloud computing service during the planning phase of an ISMS. Accordingly, our pattern can be used complementarily to the approach of Montesino et al.

Schmidt [39] presents a threat and risk-driven methodology to security requirements engineering. He

derives threat and risk models from the environment of a secure information system. This work is, therefore, complementary to ours.

Ismail et al. [23] provide a framework for analyzing the security transparency of cloud systems. Their framework considers the context of cloud systems from the conceptual view, organizational level and technical level. The concepts for analyzing security transparency on the organizational level, are defined by a meta-model. These concepts contain, among others, the identification of assets that belong to the customers of a cloud computing service. As a supplement, our pattern supports the definition of the scope and boundaries of a cloud computing service that serves as a basis for the asset identification. Hence, the above-mentioned framework could make use of our pattern.

Surridge et al. [42] present a risk management method. This method supports the automated identification of threats and controls. Our approach can provide input to such a risk assessment method. Hence, their method and our context analysis can be used to complement each other. A combination of both methods is one of the key contributions of WP7 in the context of the RestAssured project. Further detail related to the complementary methodology will be given in D7.3.

3.2 Detailed System Modelling

3.2.1 Model Hierarchy

To perform a more detailed and complete risk assememnt the individual assets of the system must be identified, and their relationships determined. Graphs provide a natural way of representing such relationships. The assets in a system represent the nodes in a graph, and the relationships between them are the edges.

At the lowest level conceptually are directed graphs. They are commonly serialised into triples, where a triple represents one edge in a graph and comprises of a subject (the source of the relationship), a property (the label of the edge or relationship type) and an object (the target of the relationship). A commonly used model to manage graphs is RDF[3], which introduces the notion of different node types and properties. An RDF dataset typically consists of an ontology (the schema) and the data, which refers to concepts defined by the ontology and can be serialised in different ways, e.g. RDF/XML or Turtle. To efficiently manage such datasets, the concept of segregated graphs has been introduced. The information is recorded by adding an extra field to each triple, effectively turning it into a quad where the last field contains the URI of the graph to which this triple belongs. The resulting collection of datasets can be treated like different tables in a relational database: they are stored separately but can be combined for queries.

Building on these technologies, our approach utilises three distinct layers that build on each other:

- **A core model**: the schema that captures the fundamental concepts, e.g. assets, roles, threats, and their relationships.
- **A domain model**: an encoding of a domain-specific knowledge base, e.g. detailed descriptions of threats and their possible control strategies.
- **A system model**: representing an actual system upon which the risk assessment is being performed.

In RestAssured we use one core model and one domain model (representing the domain addressed by the project), although the domain model is evolving in response to user requirements and the tools allow more than one domain model to be installed if necessary. Validation scenarios are modelled using a series of system models, each of which builds on one specific domain model version.

In the following sections we describe each of these in more detail.

[3] https://www.w3.org/RDF/

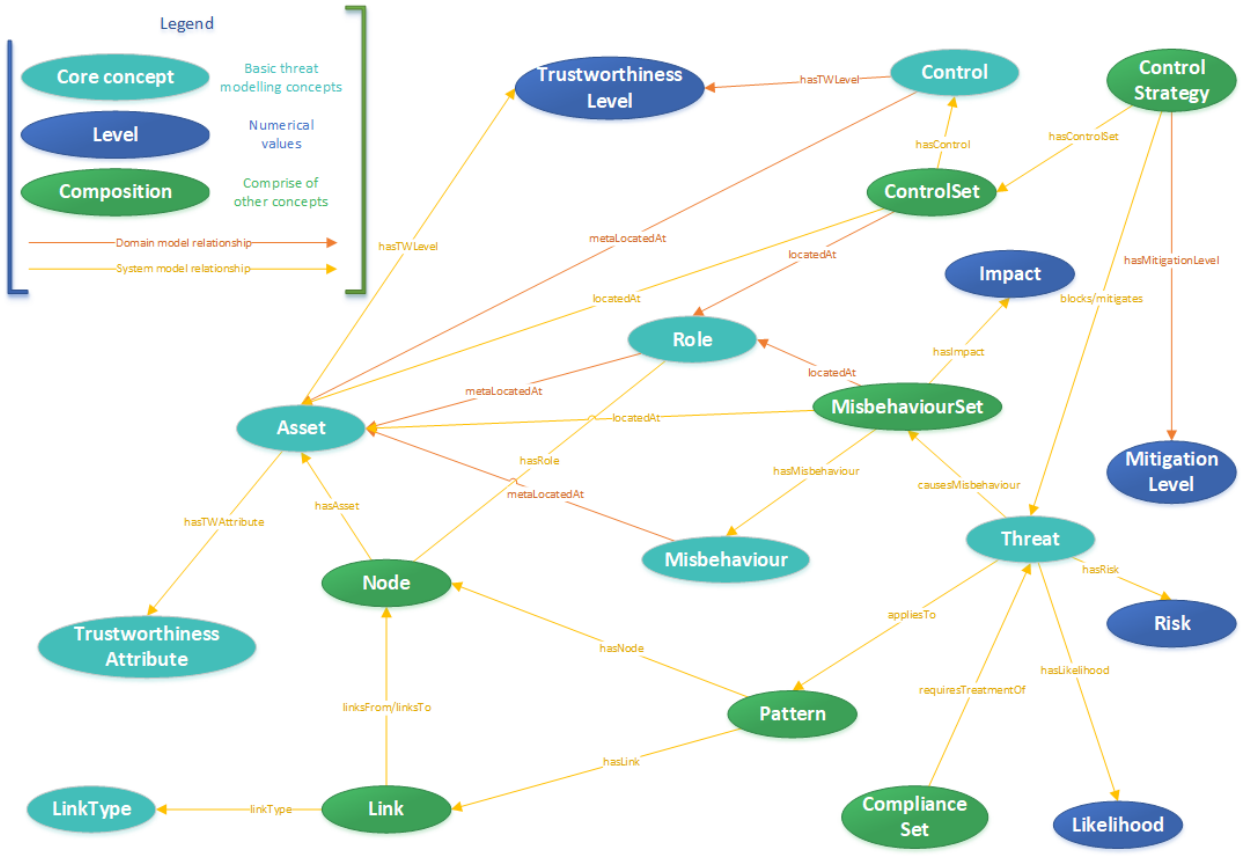

Figure 3.2: The core model (simplified version)

3.2.2 Core Model

The core model captures the fundamental concepts, and the relationships between them, that the domain models and system models build upon. A simplified version is shown in figure 3.2.

As the underlying schema for both the domain models and the system models, the core model defines a collection of concepts and relationships between them. Some of these concepts are used throughout the model stack, others are helper constructs, supporting inference of more obscure connections between these concepts. For this reason, some of the concepts have been left out of the core model overview as they appear only in the domain model but not in the validated system model.

The RestAssured core model is an enhancement of the core model from previous projects (SERSCIS, OPTET, ASSURED, 5G-ENSURE). The concepts of Asset, Role, Misbehaviour, Threat, Control etc. are carried over from this earlier work, along with relationships like:

- Asset has Role
- Threat threatens Asset
- Threat causes Misbehaviour
- Control blocks Threat

The major changes for RestAssured are the addition of *impact levels* associated with the misbehaviour of an asset, *likelihood* and *risk levels* for threats and misbehaviours, and *trustworthiness levels* associated with assets.

Each asset can have multiple trustworthiness levels, each associated with a different qualities of the asset. In a system model the system designer or risk analyst would specify the trustworthiness levels based upon their knowledge of the system itself, what they know about the provenance of the asset, and general knowledge of the threat environment.

The trustworthiness levels are then used to generate the likelihood levels of the threats, and from that the likelihood levels for the possible asset misbehaviours, for example *Loss of Availability*, or *Loss of Control*.

Finally, the system designer specifies the impact level of these asset misbehaviours based upon how important each asset is to the business or any regulatory requirements, and from this, and the likelihood level of the misbehaviours, the risk level of each asset misbehaviour is computed.

3.2.3 Domain Models

The domain model encodes domain specific knowledge. The starting point is a definition of the different possible asset classes that might exist in a given domain. These will typically always include things like hosts, processes, data, stakeholders, etc., but may also include much more specialised asset classes that are specific to a given domain.

The asset classes in a domain model form an inheritance hierarchy, so for example we can have an asset class for Human that is a subclass of the Stakeholder asset class. Inference rules, and in particular threat identification patterns (see Section refsec:threats), that apply to a parent asset class will also apply to assets from its subclasses. Some asset subclasses cannot be directly asserted by the user, either because they are base classes used only to define inherited rules or other properties, or because they are created only via inference rules (see Section 3.2.5). The asset classes from the RestAssured domain model are described in Section 3.3.3 and Figure 3.12, so will not be further elaborated in this description of modelling concepts.

Other concepts from the core model like Roles, Controls or Misbehaviours need to be specified for each domain model. For example, for each asset class the domain model also needs to define the different roles it can perform in a system model (e.g client or service, master or slave), its possible misbehaviours (e.g loss of availability), and the security controls that can be applied to it. A domain model also contains default values for trustworthiness and impact levels (see Section 3.2.6.2), though these can later be overridden by the system designer in a given system model.

The bulk of a domain model consists of a catalogue of threat patterns (see Section 3.2.6). These represent the different possible ways that assets of the classes defined in the domain model can be threatened, and the possible control strategies that can be applied to address those threats. The threat pattern catalogue therefore encodes knowledge about the threats that exist within the domain represented by the domain model. The System Security Modeller (SSM) tool uses this to generate a comprehensive catalogue of threats for each system model.

3.2.4 System Models

System models represent actual systems. They are created by using the System Security Modeller (SSM) tool from IT Innovation (see Section 4.2). System Security Modeller covers the following steps:

1. Creation by a system designer or security analyst of a model of a system in terms of its assets and their relationships to each other.

2. Automatic construction of inferred parts of the system model, needed for risk analysis but difficult for the user to specify in detail.

3. Automatic generation of a threat catalogue, along with threat causes and effects, and possible security measures.

4. Specification by the system designer or security analyst of their trust assumptions, and the impact of potential compromises of the primary assets.

5. Selection by the system designer or security analyst of security measures, and automatic evaluation of risk levels.

The system model is created by defining the assets from which it is composed, using the asset classes defined in a suitable domain model. Each system model can only use one domain model, but each domain model can be used by any number of system models. The types of relationships between the assets in a system model are also specified in the domain model.

The system model should reflect a high-level view of the system consistent with the requirements of the risk analysis. Any details needed by SSM to automatically apply security knowledge in the domain model can be automatically constructed. The system model should also represent a so-called 'sunny day' scenario in which the system behaves as the designer intends, with no security breaches. It is therefore not necessary to include potential attackers as assets, which are treated as external factors and modelled in terms of threats. Users should be represented acting in their legitimate roles, as the possibility of insider attacks is modelled in terms of trustworthiness attributes. All these aspects are discussed in the following sections.

SSM reads the domain model and offers the user a palette of available asset types to drag and drop into a visual modelling canvas. Once there, assets can be connected using a point and click procedure. SSM determines the type of relationship from the asset types being connected together, or (if more than one type of relationship is possible) provides a drop-down selection tool so the user can choose between them.

The core model therefore determines the functionality supported by SSM, the options available to a user are defined by the domain model, and the system model captures its users' knowledge of a specific system of interest. The domain model also contains the knowledge base used for automated machine inference.

3.2.5 Automated inference

Once the user has defined a system model, the machine reasoning inference rules specified in the domain model are used to achieve two things:

- to incorporate security knowledge into the system model, i.e. knowledge concerning potential threats and possible countermeasures;

- to refine and extend the model as necessary so the rules for adding security knowledge can be used effectively.

The second of these is a pre-requisite for the first. It allows SSM to add information that would be too laborious to expect users to specify it manually, or that would not occur to a user unless they had a deep understanding of security knowledge and the way it is encoded in the domain model. The rules for doing this are referred to as *construction* patterns.

For example, when a device is connected to a network, it does so via a network interface. The interface is really part of the device, but it can be attacked separately from the rest of the device, and it can play a role in defending from such attacks. This becomes evident when one considers router devices, connected to multiple networks. An attack can be made against the router from one network and not the others, overloading one interface but leaving the other interfaces (and the computational or storage capacity of the device) unaffected. To fully specify the network in such a way that these attacks can be modelled, one must include the device, the network segments (usually referred to as subnets), *and* the interfaces by which the device is connected to some subnets. Putting in all the interfaces would be a lot of work for an SSM user, and it is unnecessary because the presence of an interface can be inferred (using a construction pattern) from each connection of a device to a subnet.

Interfaces are physical and easy for users to understand, just laborious to define explicitly when their presence can be deduced. Other types of assets and relationships added by construction patterns are often far less obvious, but still important for the analysis of risks, e.g. other types of asset relationships, communication paths through networks, and the ability to make choices are all assets that can be targeted or exploited in a cyber attack.

In the first release to the RestAssured project, the focus was to incorporate the functionality for ISO 27005 risk analysis into SSM and develop a domain model containing all the information needed to support such an analysis. Very little work was done on the construction patterns, which were taken from previous projects such as FP7 OPTET.

In the latest release (in summer 2019), far more emphasis was placed on making things easier for the user by drastically expanding the use of construction rules. This was partly a response to observations during validation experiments with the first release, in which it became clear that users don't only have problems with non-obvious assets like freedom of choice. Aspects such the flow of data between processes were often not specified by users, leading to unreliable results from the risk analysis. To remedy this, iterative construction support was introduced into the SSM, allowing network and data flow paths to be inferred from a far smaller set of initial inputs, which users are far more likely to specify correctly.

Once the model has been enriched and completed through the application of construction rules, security information can be added by further machine inference rules:

- Asset attributes representing their trustworthiness properties and possible misbehaviours;
- Models of threats that could arise in the system;
- Models of security measures that could be applied to each asset, and control strategies that use combinations of these measures to counteract (some of) the identified threats.

Once these have been added, the user can specify (by overriding default settings) their assumptions about the trustworthiness of assets in their model, and the impact levels for misbehaviours. These are used as input to the automated evaluation of risk levels, as described in Section 3.2.7.

3.2.6 Threat Models

3.2.6.1 Threat patterns

Each domain model contains a catalogue of threat models that apply to that domain. Each threat class is modelled in terms of:

- **Identification criteria** The circumstances in which the threat could arise in a system, defined in terms of the presence of assets with specific relationships that would be involved in that type of threat.
- **Causes** The factors that determine how likely the threat will be.
- **Effects** The effects of the threat, i.e. the types of misbehaviour caused in one or more of the involved assets.
- **Countermeasures** The combinations of security measures required at one or more involved assets that would prevent the threat from causing these effects.

Figure 3.3 shows a typical example threat from an SSM domain model, corresponding to remote exploitation of a bug to take control of a device. The graphical notation is as follows:

- Black rectangles denote the types of related assets involved in the threat model. In this case we need a Host asset representing the device, connected to a LogicalSubnet representing the network subnet into which the attacker can send a malicious message, and the Interface between the device and the network.
- A red rectangle denotes the threat ID and the threatened asset (in this case the host).
- Red ovals represent the threat consequences in terms of asset misbehaviours, in this case loss of control over the host.
- Blue rounded rectangles represent the trustworthiness properties of these assets.
- Green ovals represent a possible control strategy, comprising a set of security measures (in this case just one) implemented at specific involved assets.

Normally control strategies are shown as separate diagrams because there may be several of them, but here we included a control just to show a mitigation along with some threat causes.

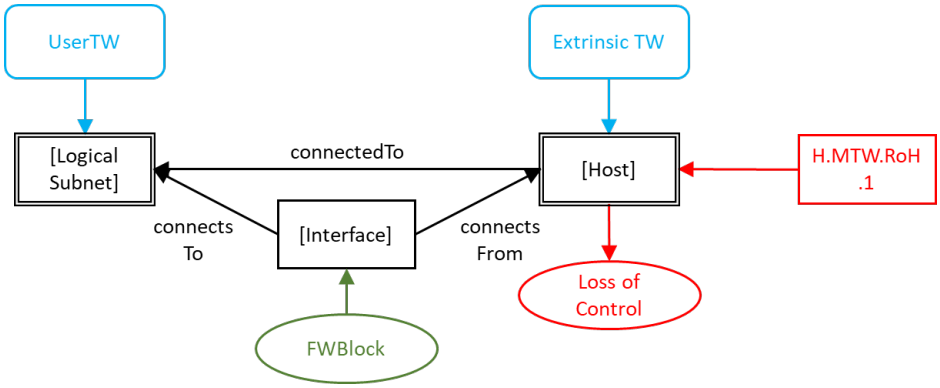

Figure 3.3: Example threat pattern and associated elements

An SSM threat model does not correspond to exploitation of a specific vulnerability as described by an entry in the Common Vulnerabilities and Exposures (CVE) catalogue[4]. It doesn't even correspond to a class of exploitable bugs as described by the Common Weaknesses Enumeration (CWE) taxonomy[5]. For example, the threat from Figure 3.3 may arise through deserialization of untrusted data (CWE-502), improper neutralisation of elements used in a command (CWE-77), argument injection or modification (CWE-88) or a range of other types of weaknesses. What matters in an SSM threat model is not the type of bug that was exploited, but the circumstances in which the threat can arise (the black elements in Figure 3.3), the causes of the threat (the blue rectangles), the consequences (the red ovals), and potential countermeasures (the green ovals in Figure 3.3).

In fact, the same weaknesses are also related to other SSM threats that have different consequences, such as the example shown in Section 3.2.6.2, where the same types of vulnerabilities could be exploited to cause a different effect, a denial of service. The mapping from CWE types to SSM threat classes is therefore not many-to-one, but many-to-many. SSM threat classes are more closely related to factors used in the Common Vulnerability Scoring System (CVSS)[6], such as the attack vector (whether the attack is physical, local or remote), the privileges needed by the attacker, the privileges gained by the attacker and/or other outcomes such as the effect on confidentiality, integrity and availability.

[4] https://cve.mitre.org/
[5] https://cwe.mitre.org/
[6] https://www.first.org/cvss/

3.2.6.2 Primary threats

Individual threats in an SSM domain model are classified as either primary, secondary or compliance threats. A primary threat is one that is caused by external factors, i.e. entities outside the modelled system but having some access to or effect on its assets, or flaws in assets that were present before entry into the system, and may cause those assets to diverge from the expected behaviour. Figure 3.4 shows a typical primary threat, very similar to the one from Figure 3.3, except this time the effect is denial of access.

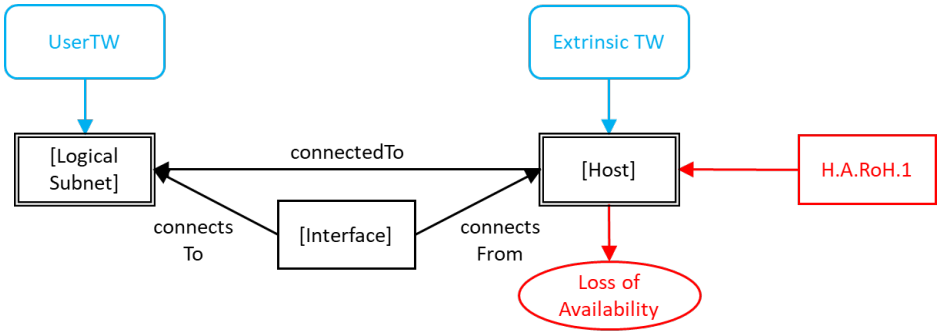

Figure 3.4: A primary threat: remote exploit to deny access to a system device

In an SSM system model, threats are autogenerated and therefore quite numerous (hundreds of individual threats even for a relatively small system model). SSM users cannot specify the likelihood of each threat in the manner proposed by ISO 27005, even if they possessed all the system and cyber security knowledge needed to decide the likelihood levels. SSM therefore computes the likelihood of threats from the likelihood of their causes. For primary threats these are determined from the trustworthiness attributes of involved assets. These attributes are known as the threat *entry points*, which are far less numerous and easier for the user to specify.

In the example from Figure 3.4, the entry points (denoted by the blue rounded rectangles) are:

- how likely it is that a malicious external actor can send messages on the subnet, represented by the user trustworthiness attribute of that subnet;
- how likely it is that the device contains an exploitable bug (i.e. a vulnerability), represented by the extrinsic trustworthiness of the asset.

Note that extrinsic trustworthiness relates to bugs that can be exploited by an external agent, not all bugs. Other types of bugs that cause problems with no external provocation are modelled via a different trustworthiness attribute.

To find the likelihood of the threat, we combine the trustworthiness levels of its entry points, then modify the result based on the security measures selected in the system model at the involved assets (see Section 3.2.6.4). The likelihood of each threat effect (in this case loss of availability at the host asset representing the attacked device) is determined from the likelihood of all threats that could cause that effect, including but not limited to this one.

3.2.6.3 Secondary threats

Figure 3.5 shows a typical secondary threat model. Secondary threats model the propagation of threat effects within or between assets. The likelihood of a secondary threat does not depend on the trustworthiness properties of involved assets, but the presence of misbehaviours. In the graphical notation used in Figures 3.3, 3.4 and 3.5 these secondary effect conditions are shown as red rounded rectangles. All the secondary effect conditions specified in the threat model must be present to trigger the secondary threat, although most secondary threats are triggered by just a single cause.

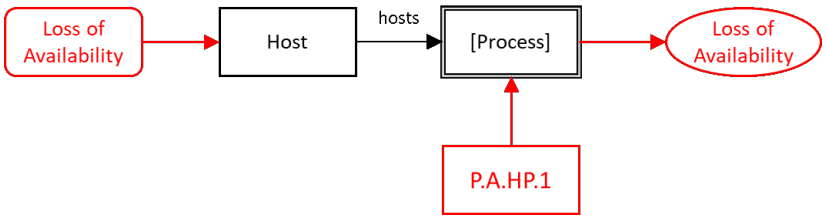

Figure 3.5: An secondary threat: loss of availability propagation

In this example, the loss of availability at a host representing a system device causes loss of availability for any process running on that device. Every secondary threat describes such a cause-and-effect mechanism whereby the effects of threats may propagate to dependent assets.

For secondary threats, the likelihood of the threat is found from the likelihood of its secondary effect conditions. The likelihood of each threat effect is still determined from the likelihood of all threats that cause each effect.

3.2.6.4 Control strategies

Control strategies represent sets of security measures implemented at involved assets that prevent a threat from arising, or (more accurately) prevent it causing effects in the involved assets. An example is shown in Figure 3.6 for the threat described by Figure 3.4.

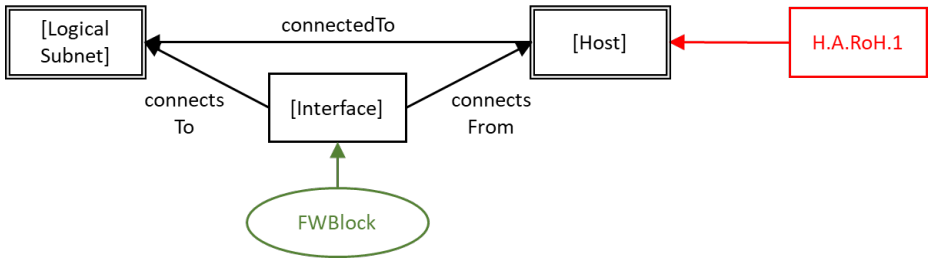

Figure 3.6: Control strategy for a remote DoS attack: block messages from the malicious attacker

Each control strategy requires the presence of one or more controls, all of which must be present for the control strategy to take effect. In Figure 3.6, only one security measure is needed: a firewall rule causing messages from the logical subnet to be dropped at the interface to the host. Note that the FWBlock control used in this case means messages should be dropped emphunless they are directed to or from a process running on the host device. This control therefore prevents attacks on the device but not on services that are supposed to run on the device and be accessed via the network.

Each threat can have more than one control strategy. Figure 3.7 shows an alternative strategy for the same threat.

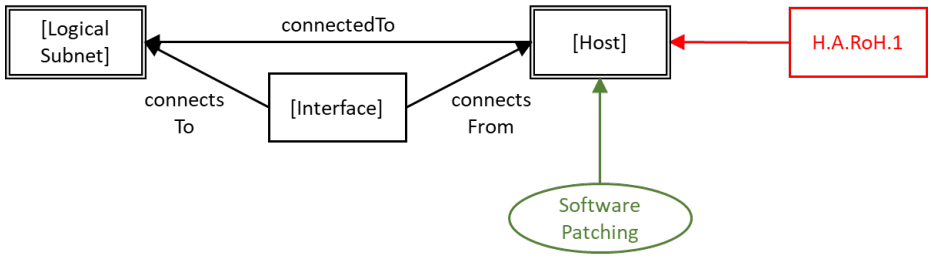

Figure 3.7: Control strategy for a remote DoS attack: patch software bugs that can cause a crash

Each control strategy has a control strength or blocking effect, expressed as a trustworthiness level. The likelihood of a threat, as determined from its entry points or secondary effect conditions, may be reduced if any control strategies are active and have a high enough blocking effect. In the current mathematical formulation, the blocking effect of a control strategy provides an upper bound on the likelihood of any threat to which it applies.

3.2.6.5 Secondary effect cascades

A secondary effect cascade is a chain of secondary threats representing the propagation of an initial threat effect (i.e. a misbehaviour) via multiple steps. Figure 3.8 shows a further secondary threat model representing the fact that if a process is not available, a data flow from that process will also cease to operate.

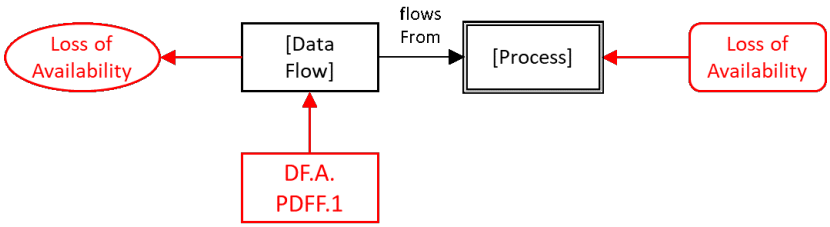

Figure 3.8: Interruption of a data flow caused by loss of availability in a process

A very simple example of a secondary effect cascade can now be constructed:

- primary threat H.A.RoH.1 (Figure 3.4)) causes a loss of availability in a host;
- this triggers secondary threat P.A.HP.1 (Figure 3.5)), representing propagation of the effect to a dependent process;
- this in turn triggers secondary threat DF.A.PDFF.1 (Figure 3.8), interrupting a data flow via that process.

Cascades involving loss of availability are very easy to understand, because the secondary threats from which they are composed represent simple asset dependencies. They are also usually simple to analyse because there isn't much one can do to prevent the propagation steps. Containment is thus difficult, and usually the best defence is to prevent the primary threat at the start of the cascade. Having said that, any asset misbehaviour (including loss of availability) can trigger multiple secondary threats, so even a simple availability cascade is usually a tree that can spread to many assets, rather than a simple chain as in this example.

One thing worth noting is that if a misbehaviour is present, any secondary threats will be automatically triggered. There is no threat agent in a secondary threat able to choose whether to propagate the effects according to the threat model. The only way to prevent an increase in misbehaviour likelihood throughout the chain of dependent assets is to contain the effects using control strategies. In the above example, if the primary cause cannot be prevented, about the only viable strategy involves the use of redundancy in devices, processes or data flows.

However, if any asset misbehaviours could be caused by a threat in a secondary effect cascade but are more likely to be caused by a threat that are not part of that cascade, then that misbehaviour (and any knock-on consequences) are not considered part of the cascade. They will be in some other secondary effect cascade, and thus linked to some other primary cause. Since control strategies only reduce the likelihood of threats, adding a control strategy for a threat in one secondary effect cascade may cause its effects to show up (but with reduced likelihood) in some other cascade.

3.2.6.6 Attack paths

Attack paths are in some ways like secondary effect cascades, and indeed the same SSM code is used to trace both. However, at attack path is constructed from primary threats, and represents a sequence of steps that could be used by a threat agent to carry out an attack against assets deep within the target system.

Figure 3.9 shows a primary threat in which an attacker with access to one subnet injects a message onto a different subnet by transmitting the message via a gateway device.

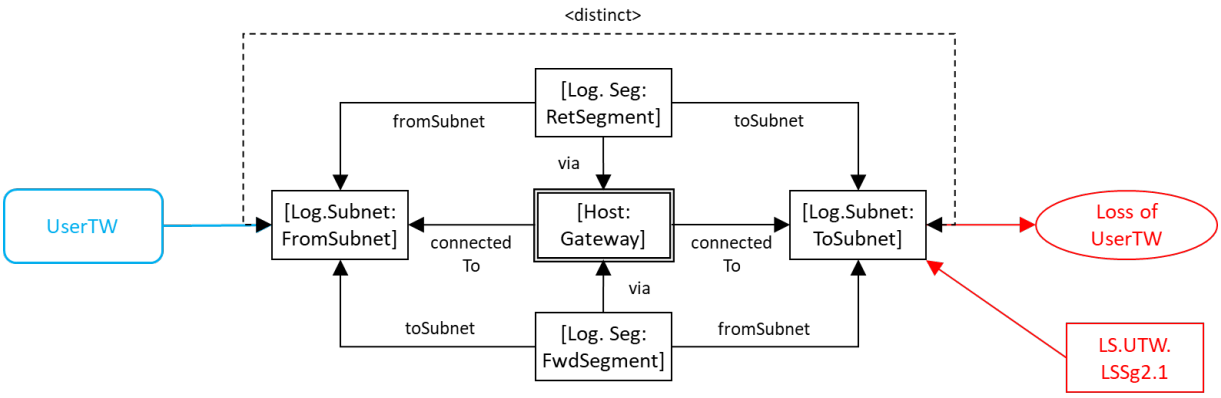

Figure 3.9: Injecting messages onto a subnet via a network router

The cause (entry point) is the user trustworthiness on the first subnet. The effect of a threat is always an asset misbehaviour, in this case a loss of user trustworthiness on the second subnet. The domain model includes several such misbehaviours that undermine an associated trustworthiness attribute on the same asset. Normally they have very low direct impact, as their effect on the system is to undermine asset trustworthiness, thus making it easier and hence more likely that other threats will occur. This is modelled by reducing the trustworthiness of an asset so it is below an upper bound determined from the likelihood of the loss of trustworthiness misbehaviour.

If a system model includes the Internet, it will normally have a very low user trustworthiness level. The likelihood of threats of the type shown in Figure 3.9 will be very high when the subnet on the left is the Internet. If no control strategy is used to prevent it, this will reduce the user trustworthiness level of the other subnet, indeed this will happen for every subnet connected via a gateway to the Internet. This in turn increases the likelihood of the same threat being launched from those subnets, eroding trustworthiness throughout the network unless controls are used to prevent it.

That also raises the likelihood of attacks on hosts connected to any of those subnets, like the one shown in Figure 3.4. A sequence of threats from Figure 3.9 enables the attacker to insert messages onto the subnet where the device is connected, and hence to launch the attack against the device itself. That in turn triggers a secondary effect cascade.

Similar effects can be achieved using threats like the one in Figure 3.10, where the attacker can access the subnet directly having gained access to a connected host. The first host in this attack path is normally a device directly connected to the Internet or running a service that is accessible from the Internet. Of course, access to any host device may also allow access to data stored or processed on the device, as well as further penetration of the network using privileges modelled in Figures 3.9 or 3.10.

Threats that model the exploitation of privileges like Figures 3.9 or 3.10 are often impossible to control, because they represent privileges used by legitimate users. A typical attack path contains a mixture of these along with some malicious compromise threats like the one shown in Figure 3.3.

Like secondary effect cascades, if a loss of trustworthiness is caused by a threat in an attack path but is more likely to be caused by a threat not in that attack path, it (and any further exploitation) is not considered part of the attack path. Instead it will appear in other attack path(s) containing the threat(s) most likely to be

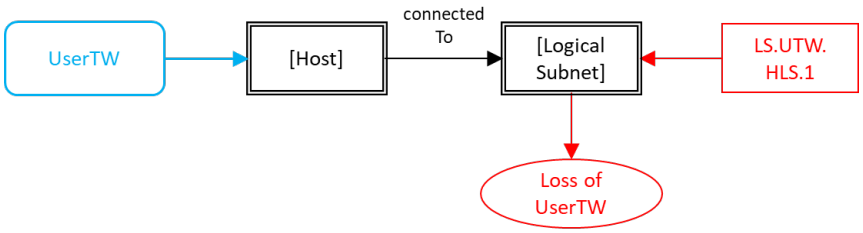

Figure 3.10: Injecting messages onto a subnet from a connected device

its cause.

Since D7.2, enhancements to the attack path analysis algorithms from the SHiELD project make it possible to include security measures in the threat patterns, to better model threats to security mechanisms. Some existing threats from RestAssured have been refactored to take advantage of this, e.g. the model for credential stuffing (part of the OWASP Top 10 A2:2017) now includes an explicit reference to the use of passwords.

3.2.6.7 Compliance threats

Primary and secondary threats are of interest in risk analysis because they have causes and effects, and model both attacks on exposed system assets and the spread of consequences through a system. The other type of threat now included in domain models is the compliance threat, an example of which was shown in Figure 3.11.

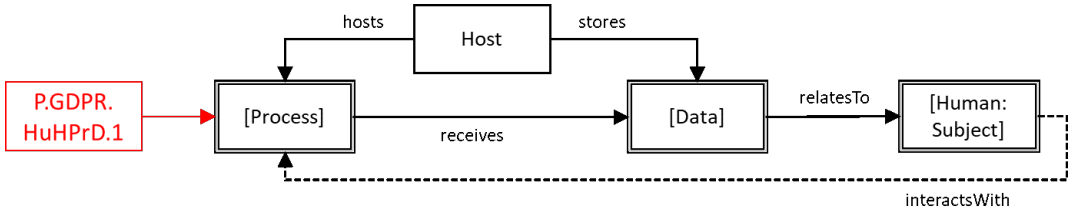

Figure 3.11: Risk of no legal basis for processing of personal data

A compliance threat does not have causes or effects, since it represents a situation that breaches some regulation, guideline or standard. One cannot talk about the likelihood of a compliance threat related to some causes (like the trustworthiness attributes or misbehaviours of involved assets). A system either complies with a modelled requirement or it does not.

The pattern of assets in a compliance threat model captures the situation in which a breach may occur. In some cases the regulatory requirement is to avoid such a situation entirely, and the corresponding threat model has no control strategies. The only way to avoid a breach of such a regulation is to alter the design of the system so that the pattern of assets does not occur, and hence the compliance threat does not arise. In other cases, the regulatory requirement specifies a situation in which certain conditions must be met. In these cases, the asset pattern captures the situation, and control strategies are used to model the conditions that must then be met. If the controls needed to fulfil a control strategy are used, the situation would be acceptable and the regulation would not be breached.

The example shown in Figure 3.14 represents a potential breach of GDPR Article 6, which states that processing personal data is prohibited unless certain conditions are met. The pattern of assets describes a process using data that relates to a natural person, i.e. a situation in which personal data is being processed and hence the system may breach Article 6. Control strategies are used to model the conditions, specified by Article 6.1(a) through 6.1(f), under which the processing of personal data is permitted. To address the threat, one must specify controls that fulfil at least one of these control strategies. For example, Article 6.1(a) says

processing is permitted if the data subject gives their consent. The corresponding control strategy models the technical or organisational measures needed to obtain this consent. Other control strategies model other conditions under which the processing is allowed, defined by Articles 6.1(b) through 6.1(f). If the threat arises in the system, it is addressed (meaning the potential breach of Article 6 does not occur) if measures required by at least one control strategy are included in the system design.

Compliance threats are allocated to one or more compliance sets, each of which represents a set of regulatory or other requirements. The threats in a compliance set represent potential breaches of the associated requirements. For a more detailed explanation of how such a set is used to model some (mainly technical) requirements imposed by the GDPR, see Section 3.3.

3.2.7 Risk Calculations

3.2.7.1 Formulation

The risk evaluation procedure uses a mathematical model to calculate the likelihood of each threat and misbehaviour. The rules used to implement the model are based on certain mathematical approximations and were originally chosen to emulate procedures recommended by ISO 27005 (notably Annex E), and to ensure that the procedure would converge within a reasonable time. Here we give the underlying model and approximations used.

Table 3.1 defines the symbols used in the rest of this section.

Symbol	Description
Ω	The set of possible likelihood levels.
A	The set of asset classes in a system model.
M	The set of possible misbehaviours of members of A.
T	The set of threat instances in a system model.
TWA	The set of trustworthiness attributes.
CS	The set of control strategies in a system model.
$DC\colon M \times A \to \mathcal{P}(T)$	$DC(m, a)$ is the set of threats that are the **direct causes** of misbehaviour m at asset a.
$EP\colon T \to \mathcal{P}(TWA \times A)$	$EP(t)$ is the set of **entry points** to the primary threat t. Recall that an entry point is a pair consisting of an asset a and a trustworthiness attribute twa of a (Section 3.2.6.2).
$SEC\colon T \to \mathcal{P}(M \times A)$	$SEC(t)$ is the set of **secondary effect conditions** of t. A secondary effect condition is a pair (m, a) consisting of a misbehaviour m of an asset a. If all the secondary effect conditions of t are active then t (may) be automatically triggered (Section 3.2.6.3).
$TCS\colon T \to \mathcal{P}(CS)$	$CS(t)$ is the set of control strategies **selected** for threat t.
$L_{(m,a)}$	The likelihood of misbehaviour m at asset a.
L_t	The likelihood of threat t.
$TW_{(twa,a)}$	The trustworthiness level of trustworthiness attribute twa at asset a.
TW_{cs}	The trustworthiness level of control strategy cs.
l_{twa}	The misbehaviour (in M) corresponding to loss of trustworthiness of trustworthiness attribute twa (Section 3.2.6.6).

Table 3.1: Risk calculation symbols and definitions

Likelihood is represented by a Likert scale containing an ordered set Ω of distinct likelihood levels ranging from Very Low to Very High. The SSM user interface displays a similar set of trustworthiness levels, but

mathematically speaking there is no separate scale for this. Trustworthiness is essentially the same scale as likelihood, but for reasons of human readability the labels used in SSM are assigned in reverse order, from Very High to Very Low. We say that likelihood is the inverse of trustworthiness, and vice versa.

In Section 3.2.6.6 we explained that the trustworthiness attributes of an asset are constrained by the likelihood of misbehaviours that undermine each trustworthiness property. This uses a direct application of the principle that trustworthiness and likelihood are inverted versions of the same scale, whereby the trustworthiness level is bounded by the inverse of the likelihood of that misbehaviour:

$$TW_{(twa,a)} \leq L^{-1}_{(l_{twa},a)}$$

The likelihood of asset a experiencing misbehaviour m is given by the likelihood of the most likely threat to *directly cause* m at a. If a direct cause is at the end of a chain of threats from some primary root cause, then its likelihood already incorporates the likelihood of that root cause:

$$L_{(m,a)} = \max\{L_t \mid t \in DC(m,a)\}$$

The likelihood of a primary threat in the absence of any control strategies is determined from the trustworthiness levels of its entry points (see Section 3.2.6.2):

$$L_t = \max\{TW_{(twa,a)} \mid (twa,a) \in EP(t)\}^{-1}$$

However, in Section 3.2.6.4 we explained that the trustworthiness level of a control strategy imposes an upper bound on the likelihood of a threat, and that the strongest selected control strategy has the dominant effect:

$$L_t \leq \max\{TW_{cs} \mid cs \in TCS(t)\}^{-1}$$

So once we take into account the presence of control strategies applicable to that threat, the likelihood of a primary threat becomes:

$$L_t = \min\big(\max\{TW_{(twa,a)} \mid (twa,a) \in EP(t)\}^{-1},\ \max\{TW_{cs} \mid cs \in TCS(t)\}^{-1}\big)$$

The likelihood of secondary threats uses a similar approach, except that secondary threats are caused not by untrustworthy behaviour at entry points, but by asset misbehaviours (see Section 3.2.6.3). So instead of taking the inverse of the highest trustworthiness of any entry point, we must use the lowest likelihood of any secondary effect condition. Ignoring control strategies this gives:

$$L_t = \min\{L_{(m,a)} \mid (m,a) \in SEC(t)\}$$

Once any relevant selected control strategies are taken into account, we get:

$$L_t = \min\big(\min\{L_{(m,a)} \mid (m,a) \in SEC(t)\},\ \max\{TW_{cs} \mid cs \in TCS(t)\}^{-1}\big)$$

Evidently these rules define a set of constraints. The likelihood of threats depends (directly for secondary threats, and indirectly for some primary threats) on the likelihood of misbehaviours, which in turn depends on the likelihood of threats. We therefore solve using an iterative procedure:

1. The likelihood of each asset misbehaviour is initialised to the lowest possible level.

2. The trustworthiness of each asset trustworthiness attribute is initialised to the user asserted level, or (if none was specified) to the default level defined in the domain model.

3. The likelihood of each primary threat is updated based on the trustworthiness of its entry points and any selected control strategies.

4. The likelihood of each secondary threat is updated based on the likelihood of its secondary effect conditions and the trustworthiness of any selected control strategies.

5. The likelihood of each misbehaviour is then updated based on the likelihood of threats that directly cause that misbehaviour.

6. The trustworthiness of each trustworthiness attribute is updated to be no higher than the inverse of the likelihood of the corresponding loss of trustworthiness misbehaviour.

7. If the likelihood of any misbehaviour changed in step (5), another iteration is performed starting from step (3).

When no further changes in misbehaviour likelihood are detected in step (7), the risk level for each misbehaviour is determined from its likelihood $L_{(m,a)}$ and impact level (which is specified as a user input).

It is also then possible to trace and delineate active attack paths and secondary effect cascades, by determining which threats (if any) produced a fall in trustworthiness of entry points $TW_{(twa,a)}$ via step (6), and which threats determine the likelihood $L_{(m,a)}$ of each misbehaviour via step (5).

3.2.7.2 Proof of convergence

The asset misbehaviour likelihoods $L_{(m,a)}$ define a function

$$L_M \colon M \times A \to \Omega$$

where Ω is the set of likelihood levels[7]. The set of such functions $\Omega^{M\times A}$ has a natural order induced by the order on Ω:

$$f \leq g \Leftrightarrow \forall (m,a) \in M \times A \,.\, f(m,a) \leq g(m,a)$$

for all $f, g \in \Omega^{M\times A}$. The minimal element in $\Omega^{M\times A}$ is the function that maps each pair (m,a) to the lowest value in Ω, and the maximal element is the function that maps each pair (m,a) to the greatest value in Ω.

The risk calculation uses an iterative procedure to find the function L_M. We start with a function in $\Omega^{M\times A}$ that maps every (valid) pair (m,a) to Very Low, and then repeatedly apply an update procedure until the generated function converges to L_M. Each application of the update prcedure corresponds to the application of a function

$$F \colon \Omega^{M\times A} \to \Omega^{M\times A},$$

and convergenece means that L_M satisfies

$$L_M = F(L_M).$$

In other words, that L_M is a fixed point of the operator F. Establishing convergence of the risk calculation therefore amounts to proving that such a fixed point exists.

To prove this, recall a function $F \colon \Omega^{M\times A} \to \Omega^{M\times A}$ is monotone if

[7] Strictly speaking not every pair $(m,a) \in M \times A$ is valid. We can address this by defining Ω to include an element $\perp$ representing zero likelihood in addition to the levels in Table 3.3.

$$f \leq g \Rightarrow F(f) \leq F(g)$$

for all $f, g \in \Omega^{M \times A}$. Such a monotone F generates a sequence

$$f \leq F(f) \leq F^2(f) \leq F^3(f) \leq \ldots$$

for any $f \in \Omega^{M \times A}$ such that $f \leq F(f)$. Now, the set $\Omega^{M \times A}$ is finite, since A, M, and Ω are all finite. Therefore there exists an n such that $F^n(f) = F^{n+1}(f)$, and $F^n(f)$ is a fixed point of F.

For the risk calculation we therefore need to show:

1. That the update procedure is a monotone function $F \colon \Omega^{M \times A} \to \Omega^{M \times A}$.
2. That the initial choice $L_M^{\#}$ for L_M satisfies $L_M^{\#} \leq F(L_M^{\#})$.

Work is ongoing to establish this. However two things can be noted:

1. The update procedure employs certain approximations (see below) and these affect whether the procedure is monotone.
2. The fixed point may not be unique:
   (a) Different choices of $L_M^{\#}$ may lead to different fixed points.
   (b) Moreover, for any monotone F as above, if one starts with $F(L_M^{\#}) \leq L_M^{\#}$ for some $L_M^{\#}$, then a decreasing sequence is generated, that for the same reasons above, must converge to a fixed point.

Therefore **which** fixed point (if there is more than one) is the correct one? The risk calculation currently employed by SSM (probably) computes the **least** fixed point, though perhaps the greatest fixed point[8] is a more appropriate choice.

3.2.7.3 Physical interpretation

As mentioned above, the risk calculation employs certain approximations during the update procedure. These simplify the calculation and result in a procedure that is monotone, and thus one that converges. The approximations we use include:

1. For each (m, a) we approximate $L_{(m,a)}$ by the maximum value of L_t for all threats t that are direct causes of (m, a).
2. For each secondary threat t we approximate L_t by the minimum value of $L_{(m,a)}$ for all the secondary effect conditions (m, a) of t.

The first assumption is based on the idea that the distinct levels defined in the scale of likelihood are widely separated, to the extent that larger values dominate the calculation. More precisely, if we consider each likelihood level represents a statistical probability in the range [0, 1], then we assume that these probabilities for adjacent levels differ by a factor N, where N is a sufficiently large number.

Now suppose we have three adjacent likelihood levels $x < y < z$, and the most likely threat that directly causes misbehaviour (m, a) has likelihood y. If only one threat is at level y, and the next $n << N$ have

[8] As $\Omega^{M \times A}$ is a complete lattice, by Tarski's Fixed Point Theorem the set of fixed points of F is a complete lattice, and thus has both least and greatest elements.

likelihood level x, then because $n << N$ those n threats contribute little compared to the one threat at level y. We conclude that the first threat dominates and the likelihood level for the misbehaviour is y. On the other hand, if there are $n << N$ threats at level y, and none at the level z above y, then because $n << N$ the threats at level y will not (even together) raise the chances of the misbehaviour enough to place it at level z, so still the likelihood level for the misbehaviour is y.

This follows from the fact that we only need one threat to cause the misbehaviour, so if we have two causes A and B with probability $P(A)$ and $P(B)$, the probability of the misbehaviour is given by the probability of the disjunction of causes. If A and B are statistically independent, that is given by:

$$P(A \vee B) = P(A) + P(B) - P(A \wedge B),$$

However, there is a caveat to this. The approximation is only valid provided the number of direct causes is much less than N. In fact it turns out that this is also not enough, as probability can also accumulate in chains of threats, so it is likely that what really matters is that the total number of *statistically independent* threats in the *causal tree* of (m, a) must be much less than N. In practice the number of distinct causal chains (attack paths or secondary threat cascades) can be large, with tens or even hundreds of chains leading to a given outcome. This implies that in some cases it is may not enough to place the likelihood levels one order of magnitude apart, instead N may need to be 100, or maybe more. However, many of these chains overlap so they are not statistically independent, so the above analysis is not strictly valid, and the approximations are safer than they first appear.

For secondary threat causation we have a different problem. Suppose a secondary threat t has two secondary effect conditions A and B (i.e. two misbehaviours that together trigger the secondary threat). If A and B are statistically independent with probability $P(A)$ and $P(B)$, then since we need both to trigger the threat, the probability of the threat is that of the conjunction of causes:

$$P(A \wedge B) = P(A)P(B).$$

If we now assume A and B have different likelihoods, whose probabilities are separated by a power of $(1/N)$, we can say $P(A) = p/N^m$ and $P(B) = p/N^n$, where $m < n$ and hence $P(B) < P(A)$. The rules used above would say the likelihood of the secondary threat should equal that of the least likely cause, that is to say $L_t = P(B) = p/N^n$. But the conjunction of causes should have likelihood $P(A \wedge B) = p^2/N^{m+n}$. Unless $P(A)$ is close to 1 (meaning the more likely cause A is nigh on certain), this is simply not the case.

This situation is quite different from that of misbehaviour causation, because we need a conjunction not a disjunction between causes. SSM may therefore overestimate the likelihood of some threats. However this is also not a problem in the RestAssured project, as secondary effect threats in the RestAssured domain model only have one secondary effect condition, i.e. one direct cause. Even if that were not the case, the problem would only arise if there were multiple causes that are statistically independent, and here too the fact that secondary effect cascades are numerous and overlap to a considerable degree means that if a secondary threat does have multiple causes, they are likely to be at least somewhat correlated.

The conclusion is that the approximations used to simplify the computation and ensure convergence can be relied upon, provided successive levels in the likelihood/trustworthiness scale represent probabilities an order of magnitude apart, and the domain model does not use secondary threats caused by the conjunction of multiple triggering misbehaviours.

3.2.7.4 User inputs

The automated risk evaluation procedure implemented by SSM uses the ISO 27005 methodology:

- the likelihood of each identified class of threat is calculated from asset trustworthiness levels or misbehaviour likelihoods, taking into account any security controls;

- the likelihood each potential asset misbehaviour is found from the likelihood of the threat(s) that could cause it;
- the risk level for each asset misbehaviour is found from its likelihood and impact.

To use the risk evaluation feature, the SSM user must first specify the trustworthiness levels of assets and the impact levels of asset misbehaviours.

During the risk evaluation, SSM doesn't just calculate threat and misbehaviour likelihood. It also traces secondary effect cascades and attack paths, i.e. chains of threats that represent the propagation of misbehaviour through the system, or that could be used by an attacker to gain access to an initially unexposed part of the system. Results from these calculations are fed back into the estimates of threat likelihood as discussed in Sections 3.2.6.5 and 3.2.6.6 respectively. This greatly simplifies the problem of specifying trustworthiness and impact levels.

ISO 27005 provides guidance on how to determine the impact of an asset compromise in Annex B. This defines *primary* assets as those that have significance beyond the IT system of which they are part. Usually the primary assets consist of certain data and processes whose compromise would have immediate business consequences. The impact of such a compromise should be based on those consequences.

ISO 27005 defines *secondary* assets as those needed to support the primary assets, e.g. if a data set is a primary asset, then the device used to store that data will be one of the secondary assets. By this definition, almost all networks and devices, software processes and users in a system will be secondary assets. ISO 27005 recommends that the impact level for compromise of a secondary asset should be based on factors like the cost of recovery or replacement, and the impact on any dependent assets. It notes that the more dependencies there are on an asset, the higher the impact of a compromise will be, implying that a dependency analysis is required to set realistic impact levels. Suggestions include using simulations or the effects of previous or similar incidents to determine how this should be done.

In an SSM system model, each type of compromise is represented as a potential asset misbehaviour. These are auto-generated by SSM with a default impact level specified in the domain model for each type of asset. A few asset types like 'sensitive data' which are normally primary assets have moderate to high default impact levels, but the rest are mostly very low. Asset dependencies do not need to be considered because they are in effect simulated by the automated analysis of attack paths and secondary effect cascades. Because of this, the low default impact levels are acceptable for secondary assets, and the SSM user usually just needs to check and adjust impact levels at primary assets.

In ISO 27005, guidance on how to determine likelihood is very sparse. The standard says this should take account of how often threats occur and how easily vulnerabilities can be exploited, considering:

- experience and applicable statistics for threat likelihood;
- the motivation, capabilities and resources of attackers, and their perception of the attractiveness and vulnerability (to them) of the threatened assets;
- the proximity of accidental threat sources, and factors that may influence malfunctions and human errors;
- vulnerabilities, both individually and in aggregation; and
- existing controls and how effectively they reduce vulnerabilities.

Annex E recommends that the likelihood of a specific threat (involving exploitation of a specific vulnerability) can be estimated from:

- the attractiveness of the threatened asset (or threat consequences) to an attacker;

- the ease with which exploiting a vulnerability of the asset could be converted into a reward for the attacker;
- the technical capabilities of the attacker; and
- the susceptibility of the vulnerability to exploitation by technical or non-technical means.

All these statements are clearly reasonable, but they are also very general and do not provide much insight into the likelihood of a specific threat. They are also somewhat interdependent and so don't provide a very usable set of criteria for estimating threat likelihood in general. Annex E goes on to provide examples of how to combine likelihood and impact to obtain a risk level but adds to the confusion by giving examples in which factors like 'ease of exploitation' (strictly related to likelihood only) are incorporated in the final risk level assessment stage.

In an SSM system model, the likelihood for each primary threat is found from the trustworthiness of their entry points, and for each secondary threat from the likelihood of triggering misbehaviours, in both cases moderated by the effect of security measures indicated in the system model. The user therefore only needs to specify trustworthiness levels for assets in the model.

Trustworthiness attributes are (like misbehaviours) auto-generated for each system asset, and default trustworthiness levels set based on the type of each asset. For a few asset types representing public spaces or networks, default trustworthiness levels are very low, reflecting the fact that these assets can be used by anybody including malicious attackers. Asset types that routinely succumb to attacks have intermediate default trustworthiness levels, e.g. devices are considered only moderately free of exploitable vulnerabilities in the absence of software patching or other security measures. The remaining default trustworthiness levels are very high.

SSM adjusts the trustworthiness levels of each asset as part of its analysis of attack paths, in effect simulating the effect of asset dependencies on system trustworthiness. The user can ignore these dependencies and focus on how trustworthy each asset should be in the 'sunny day' scenario captured by their system model. The (generally high) default levels are therefore mostly acceptable, especially for secondary assets. Some may need to be adjusted where they represent significant sources of risk, e.g. user roles that may be occupied by less trustworthy individuals, or assets that are not dedicated and may be accessed by external attackers.

Because trustworthiness attributes represent external factors not captured in the system model, they can also be used to limit the scope of a system model. For example, suppose a model of an organisation's network includes a WiFi network that is used to allow visitors to access the Internet. One option would be to include the visitor role in the model and specify an intrinsic trustworthiness level representing the possibility that someone with the privileges of a visitor (including access to that WiFi network) may in fact be a malicious intruder. Another option could be to leave out the visitor role, but treat the WiFi network as a non-dedicated asset with users who are not modelled as part of the system, and set the user trustworthiness of the WiFi network accordingly.

In summary, the ISO 27005 risk evaluation procedure supported by SSM involves:

- review and adjustment of default trustworthiness levels, based on the extent to which the asset is used for other purposes than those included in the system model;
- review and adjustment of default impact levels for primary assets;
- automated calculation from these inputs of the likelihood of threats and asset misbehaviours;
- automated calculation of risk levels based on the likelihood and impact of asset misbehaviours.

The user can then iteratively specify security measures to reduce the likelihood of certain threats, and update the risk level calculation, until the residual risk levels are acceptable.

3.3 The RestAssured Domain Model

3.3.1 Domain model overview

The RestAssured domain model is designed to facilitate identification and evaluation of data protection risks in cloud environments. Both SSM itself and the domain model used have been developed in RestAssured and in the H2020 SHiELD project, whose focus is on end-to-end data protection risk analysis in cross-border health care scenarios. Risks can only be evaluated at the level of an application, where the impact on stakeholders (including data subjects) can be assessed. As a result, even though SHiELD does not address cloud environments, the bulk of the domain models developed in each project also apply to the other, and most of the work done has been shared with both project consortia.

The main developments since RestAssured D7.1 are shown in Table 3.2.

What	Project	Note
GDPR requirements encoding	Mainly SHiELD	SHiELD is mainly concerned with cross-border scenarios in which the GDPR provides a common regulatory baseline. However, SHiELD focuses on health data transfer and doesn't consider broader scenarios, so some additional work was done in RestAssured, notably related to the use of biometric data.
Introducing cloud resource models	RestAssured	RestAssured is mainly concerned with data protection in the cloud, so although resourcing plays a small role in business risk evaluation, a basic model of cloud resourcing strategies was added.
Refactoring physical attack threat models	SHiELD	Refactored to separate different stages of a physical attacks. This refinement was needed in SHiELD in order to capture the effect of control strategies using security technology developed in SHiELD. It was convenient to use the same refactored threats in RestAssured.
Data flow models	RestAssured	Data flow assets and enriched process-data relationships were first introduced in RestAssured to improve alignment with the data lifecycle model from WP6.
Data flow construction	Both	SHiELD added construction rules to create data flow assets to simplify models of compliance requirements in cross border health data transfers. RestAssured modified these rules to use iterative data flow construction, so users would not need to specify process-data relationships at intermediate services, which users may overlook.
Data flow refinement	RestAssured	Since D7.2, the data flow assets and construction rules have been further refined to distinguish between process-to-process and end-to-end flows of data, and threat models modified to use these distinctions, allowing more efficient threat models that take account of controls like end-to-end encryption, and helping to address runtime mapping issues (see below).
Support for attack path tracing	RestAssured	Added to simplify the process of identifying threats that require countermeasures, in response to feedback from RestAssured validation partners.
Attack path extensions	SHiELD	Since D7.2, improvements have been made to the SSM algorithms for analysing attack paths, allowing simplification of models in the knowledge base representing threats to security mechanisms.

What	Project	Note
OWASP Top 10 threats added	RestAssured	Added in response to OCC's request to make it easier to see how this specific set of threats (of great interest to their customers) are addressed.
Refactoring remote exploit threat models	RestAssured	Done to reduce the number of threats in large models with many devices and subnets, which became far too large in one OCC model.
Runtime mapping support	RestAssured	Introducing extra asset classes corresponding to nodes in the runtime system model representation, including generic web and database servers, stored data sets, and the improved models of data flows.
Runtime adaptation as a security measure RestAssured	Added security control classes representing the effect of runtime adaptations.	

Table 3.2: Domain model developments since RestAssured D7.1

Because of the overlaps, some material included here is common to SHiELD D4.4 [38], which describes a software and domain model release to SHiELD partners with the same due date as D7.2. As far as possible, material describing work carried out in SHiELD is summarised, while that done in RestAssured is described in more detail.

3.3.2 Risk evaluation scales

To support the evaluation of risks in accordance with ISO 27005, SSM needs four scales representing likelihood (of threats and misbehaviours), trustworthiness (of assets), impact (of misbehaviours and indirectly, threats) and risk. These are defined as part of the domain model, so are included here for completeness, although they were previously described in Deliverable D7.1.

As noted in Section 3.2.7.1, the likelihood and trustworthiness scales are really two views of the same thing. The current domain model uses a 5-level Likert scale for both, as shown in Table 3.3. Likelihood levels are associated with threats and asset misbehaviours, and are always computed by SSM as discussed in Section 3.2.7.1. Trustworthiness levels are associated with asset trustworthiness attributes, and these are specified as user inputs, but may then be modified during the risk level calculation.

The impact scale represents the effect on business functions and stakeholders should an asset be in some way compromised. A 5-point scale is also used for this, as shown in Table 3.4. Impact levels are associated with asset misbehaviours. They are user inputs that are used but not changed in the risk level calculations.

ISO 27005 stipulates that risk is a function of likelihood and impact. The RestAssured domain model uses a 5-point scale also for risk levels, as shown in Table 3.5:

ISO 27005 specifies that risk is a function of likelihood and impact, but does not specify what that function should be. In RestAssured we have adopted the mapping in Table 3.6. At present this look up table is encoded within the SSM software, but because it depends on the above scales, it will at some stage also become part of the domain model and is included here for completeness.

Likelihood	Description	Trustworthiness
Very High	Something will definitely go wrong if the possibility exists even only for a short time.	Very Low
High	Something is likely to go wrong if the possibility exists even only for a short time. Something will definitely go wrong if the possibility exists for very long.	Low
Medium	Something is likely to go wrong if the possibility exists for very long, but not otherwise.	Medium
Low	It is unlikely that anything will go wrong without very prolonged exposure to the possibility.	High
Very Low	It is unlikely that anything will ever go wrong.	Very High

Table 3.3: Likelihood and trustworthiness levels in the RestAssured domain model

Impact	Description
Very High	This misbehaviour in this asset will be fatal to key business interests. Very few misbehaviour sets will be at this level, but those that are must be prevented.
High	This misbehaviour in this asset has a serious impact on the business, and will be fatal if not addressed quickly.
Medium	This misbehaviour in this asset can be tolerated for a short time, but will become serious if not addressed quickly.
Low	This misbehaviour in this asset can be tolerated, but it does degrade business function or efficiency.
Very Low	This misbehaviour in this asset has negligible impact on the business.

Table 3.4: Misbehaviour impact levels in the RestAssured domain model

Risk level	Description
Very High	Cannot be allowed to arise.
High	Control measures should be introduced immediately, by shutting down parts of the system if necessary.
Medium	Can be tolerated for a short time while measures are introduced to address it.
Low	Can be tolerated for a longer period, or indefinitely if the cost of further risk reduction is very high.
Very Low	Risk can be accepted.

Table 3.5: Risk levels in the RestAssured domain model

Impact	Likelihood				
	Very Low	**Low**	**Medium**	**High**	**Very High**
Very Low	Very Low	Very Low	Very Low	Low	Low
Low	Very Low	Very Low	Low	Low	Medium
Medium	Very Low	Low	Medium	High	High
High	Low	Medium	High	Very High	Very High
Very High	Low	Medium	High	Very High	Very High

Table 3.6: Mapping misbehaviour likelihood and impact to risk level

Note that this table is not symmetric, since low likelihood, high impact events are normally considered to pose a higher risk than high likelihood, low impact events.

3.3.3 Asset classes and relationships

The asset classes currently used in the RestAssured domain model cover the following broad asset types:

- **Data assets**: used to represent the primary assets to be protected, and (where needed) secondary assets used for configuration and control purposes. Subclasses are used to represent special categories of data that are subject to different regulatory requirements under the GDPR. See Section 3.3.10.3 for details.
- **Processes**: represent running applications that may exchange and process data. The DB, WebClient and MailClient classes can be involved in additional classes of threats.
- **Hosts**: represent physical or virtual devices that may store data, run processes, and communicate over networks. The physical host subclasses are mainly of interest in SHiELD but virtual host classes were added in RestAssured D7.2 to model cloud resourcing and virtualisation based security.
- **Logical Subnets**: represent locally connected subnets to which Hosts may be connected. The VLAN class was added for RestAssured D7.2. Classes like the Internet differ in having much lower default trustworthiness properties.
- **Spaces**: represent physical locations where devices may be located, and subnets (notably radio subnets) may be accessed. The DataCentre was added for RestAssured D7.2, a space containing devices and networks that are not modelled explicitly, but can be involved in certain threats.
- **Stakeholders**: represent humans (system users and data subjects) and organisations involved in the operation of other assets.
- **Other assets**: currently used only to represent regulatory context via the Jurisdiction class.

The asset class hierarchy is shown in Figure 3.12. The shading indicates that an asset class can be asserted by the user, i.e. used to build a model of their system within SSM. Some cannot be asserted, usually because they are base classes used to encode rules or properties that apply to subclasses but are not themselves fully defined. For example, the Space asset is involved in threats representing attacks that require physical access to a device, but it has no default trustworthiness properties because they can only be defined for public and private spaces. SSM users therefore must choose which type of space they wish to use in a system model.

There are also some non-assertable asset classes not shown in the diagram which are only ever created by construction rules. One of these is the Interface class described in Section 3.2.5.

Asset relationships cover a wide range of situations. The basic relationships used are:

- between Hosts and Logical Subnets, describing network provisioning and connectivity;
- between Hosts and Processes or Data, describing what is running or stored on each connected device;
- between Processes and Data, describing how and where data is processed;
- between Processes, describing client-service interactions which include data transfers;
- between Stakeholders and Hosts, defining who is responsible for managing or operating devices;
- between Humans and Processes, saying which users interact directly with which processes;
- between Hosts and Spaces, describing the location of devices so their physical security to be modelled;

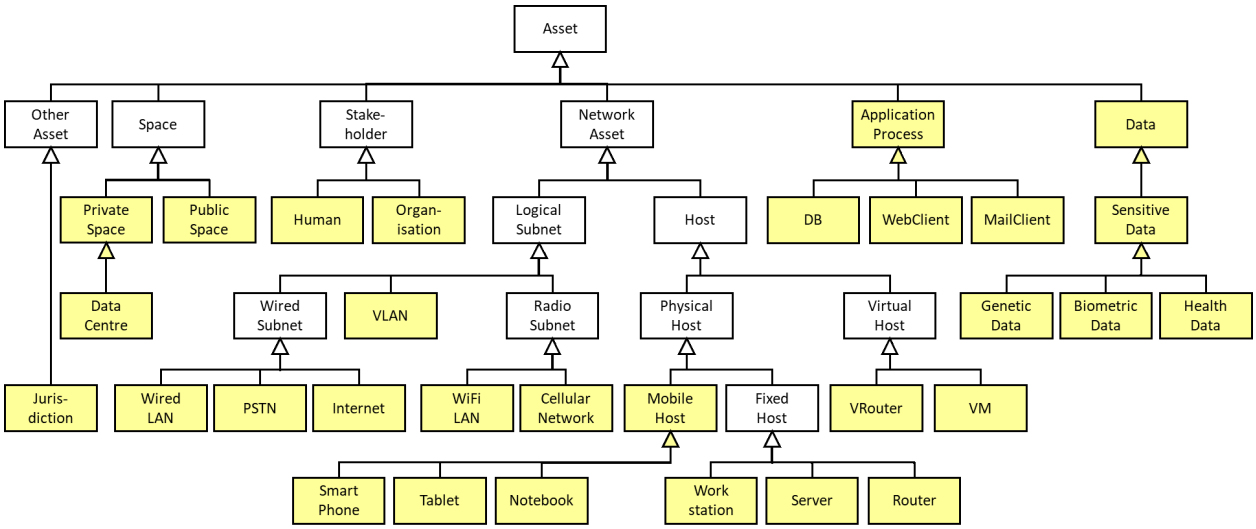

Figure 3.12: The Asset class hierarchy

- between Hosts and Jurisdictions, indicating the regulations applying to each Host its processes or data.

The main changes between D7.1 and D7.2 concern the relationships between Processes and Data. Here, the original relationship was just 'uses', but this has now been expanded into a much richer set to better capture the Create, Read, Update and Delete (CRUD) lifecycle of data used in WP6. This is explained in Table 3.7.

	Process *ignores* data value (no read or write access)	Process *uses* data (has read access)	Process *creates* data (has write access)	Process *changes* data (read/write access)
Process has no dependency on data	**serves**: Process allows other processes to read and write data which it stores.	**receives**: Process uses values but does not fail if they aren't available.	**creates**: Process creates the data so doesn't depend on getting them from elsewhere.	**amends**: Process changes data values but doesn't fail if the data is not available.
Process availability depends on being able to access data	N/A	**reads**: Process uses values and fails if they are not available.	N/A	**updates**: Process changes the data values and fails if not available.

Table 3.7: Process-Data relationships

Initially, these relationship types were specified for every process that touches data, allowing threats to be introduced for data servers, data consumers and data intermediaries, taking note of whether the process needed to be able to read or alter the data and hence whether threats might be blocked by encrypting or at least write protecting the data. Later, a data flow asset was introduced for the SHiELD project, and its presence deduced from the above relationships. This asset was subsequently adopted for the RestAssured domain model, and the inference of data flows improved by using iterative construction rules to determine their existence and direction from far fewer asserted facts. This makes defining a system model easier and less error prone for the SSM user.

3.3.4 Mapping to runtime model concepts

Since D7.2, further changes have been made in the assets and relationships specified in the knowledge base to simplify the process of mapping between the runtime system models from WP5 and the models produced by SSM.

The main issue faced was that SSM models represent the structure of a system in relation to its security requirements and properties, whereas the runtime models from WP5 reflect the way the system is implemented. A good example of these differences is the representation of a website. In the runtime system model, a Website is hosted by a Webserver, which runs on a Compute node. This reflects the fact that there is a Webserver (like Apache) that handles HTTP requests, and the Website itself is data that the Webserver uses to determine its response. In an SSM model, a Website is a process whose behaviour depends on the website content and server, which is hosted by a computing and storage device. The SSM model did not use a separate asset to represent the HTTP server devoid of any content, as normally the target for any security policy will be a web resource, i.e. a combination of HTTP server plus content. Indeed, when using Apache web server, access control is normally embedded (via the .htaccess mechanism) in the content making up a website.

It is possible to map between the runtime model and an SSM model by ignoring the Webserver nodes in the runtime system model. However, it simplifies the mapping algorithm if there is an asset in the SSM model for each relevant node in the runtime system model. For this reason, one asset class was renamed, one was redefined and three were added:

- Webserver is a new class introduced to represent the HTTP server devoid of any website content.
- DBMS is a new class used to represent a database management system devoid of any database content.
- DataSet is a new class added to represent a storable data asset, which in the runtime model contains data records that represent the data itself.
- DataStep is the new name given to a data transfer between processes that communicate directly with each other, previously referred to as a DataFlow.
- DataFlow has been redefined to represent end-to-end flows of data between source and destination processes, which may be separated by a number of intermediaries.

SSM itself is agnostic as to the content of its knowledge base, so it treats all assets in the same way. The knowledge base determines how each asset class is interpreted. None of the assets listed above can be specified as components of the modelled system by an SSM user. They are all inserted during the SSM model validation procedure based on inference rules that are also defined in the knowledge base (see Section 3.3.9). The first three are only used by the RRE mapping algorithms, and are not subject to any threats defined in the knowledge base, as threats will always target the associated website, database or data assets.

The original DataFlow asset class, now renamed DataStep was subject to some threats which have now been redefined. Some of these threats still target a DataStep asset (a direct process-to-process data transfer), while others now apply only to an end-to-end DataFlow. In most cases, a threat to confidentiality or integrity in a DataStep can be countered using end-to-end encryption for the DataFlow. Threats to DataStep availability usually create a secondary effect cascade that threatens the availability of the end-to-end DataFlow.

3.3.5 Trustworthiness attributes

The main trustworthiness attributes were introduced in D7.1:

- **Intrinsic TW**: free of internal motives or flaws that cause the asset to misbehave.

- **Extrinsic TW**: free of vulnerabilities that could allow others to make the asset misbehave.
- **Management TW**: under the control of the responsible parties, i.e. so nobody can change the asset without authorization.
- **User TW**: having users that won't misuse the asset, either for malicious attacks or due to user errors or accidents.

Note that the last two of these should consider everyone with management or user rights, not just those who have roles in the modelled system. This is one of the main ways in which an SSM system model can capture externalities. In most modelled business systems or applications, at least some assets are also used for other purposes and by external users, so this is an important feature of the domain model.

Some changes and additions were added for D7.2:

- **Authenticity**: the extent to which the asset is what it claims to be.
- **Health**: freedom from self-replicating malware.

Authenticity was used in D7.1, but at that time it was limited to being operated by the claimed stakeholder and applied mainly to processes and devices. Now it is more general, applies to more assets including communication channels and data flows, and represents the extent to which the asset is not involved in any form of deception.

Health was added for D7.2 because malware propagation often uses the same methods as human attackers, and these were already modelled as primary threats. At present, secondary threat likelihood depends only on misbehaviour likelihood, but worm propagation (in particular) should also depend on the trustworthiness attributes of the target device since the worm will normally propagate by exploiting vulnerabilities. The work around was to model worm propagation as a primary threat, which meant a trustworthiness attribute was needed as an entry point to capture the presence (or strictly absence) of infection with self-propagating malware.

3.3.6 Misbehaviour types

Misbehaviour types have been overhauled since D7.1, although many were introduced prior to that. The main changes relate to new asset classes representing data flows, and related assets such as communication channels. The main classes of asset misbehaviour that have been in use for a long time are:

- **Loss of Confidentiality**: applies to data, and now also data flows and communication channels, and signifies leaking of information.
- **Loss of Integrity**: applies to data, and now also data flows, and signifies that information has been altered, not in order to deceive, but in a way that will cause a fault when the data is used.
- **Loss of Availability**: applies to Host, Process, Data and Space assets, signifies that the asset can no longer be used by authorised users.
- **Overloaded**: applies to Process, Host and Logical Subnet assets, signifies that the load on the asset is more than expected or intended.
- **Unreliable**: applies to Process and Host assets, signifies that the asset is performing its functions incorrectly on a significant fraction of occasions when those functions are used.

Conditions like loss of integrity and availability normally persist, but dependent processes don't use the same assets in the same way every time. The persistent cause is thus unlikely to produce a persistent secondary effect. The Unreliable misbehaviour was introduced to handle this case, where things will go wrong but less often and in different ways than a loss of integrity or availability. Later, a primary threat model was added to describe the effects of an intermittent software bug.

The following misbehaviours are also included in the model, but their effect is to undermine trustworthiness attributes, so they normally aren't considered to have a significant direct impact on the affected asset:

- **Loss of IntrinsicTW**: applies to Host, Process and Stakeholder assets, undermines TW attribute Intrinsic TW, and currently used only in stakeholder on stakeholder threats.
- **Loss of ExtrinsicTW**: applies to Host, Process and Stakeholder assets, undermines TW attribute Extrinsic TW, but currently not used in any threat.
- **Loss of Control**: applies to Host, Process, Logical Subnet and Space assets, means an unauthorised party has gained control of the asset, undermines TW attribute Management TW.
- **Loss of UserTW**: applies to Host, Process, Logical Subnet and Space assets, means an unauthorised party has gained access to the asset, undermines TW attribute User TW.
- **Loss of Authenticity**: applies to data, and now also data flows and communication channels, signifies alteration or subversion in order to deceive, undermines TW attribute Authenticity.
- **Infected**: applies to Host and Process infested with malware, undermines TW attribute Health.

The general principle when defining Trustworthiness Attributes and Misbehaviour types is to try to keep the number of distinct types as small as possible. Few additions were therefore made for D7.2, mainly to allow better modelling of attack paths and secondary effect cascades, and only after very careful analysis to check that the new types really were needed.

3.3.7 Control types

Control classes have been more extensively updated since D7.1 and are also far more numerous. There are three main drivers for this proliferation:

- Threats are classified based on the circumstances in which they are relevant, their causes and effects, and the control strategies that may be used to block them. When a new threat is identified, it is often a variant of an old threat that bypasses one of its control strategies. When the new threat is added, a variant of the control strategy is also added, using variants of the control types.
- Developers of security technologies want to see their outputs included in the model. RestAssured security technologies are no exception since finding out where they are needed is one of the goals of the risk model. This also tends to produce new threat classes, to capture situations where generic security measures are not effective, and only RestAssured control variants will do.
- Users want to see controls they are familiar with, and often complain when they are missing. This favours many specific control types representing different ways to implement a control concept, rather than a single generic control type representing the need but not the means for implementation.

Here we give only the main groups of control types included in the RestAssured domain model, highlighting individual control types only where they correspond to RestAssured technologies from WP4 and WP6. The main areas covered are:

- **Authenticators**: means by which assets can prove their identity or that of their operator, or their role and privileges. Now covers passwords, digital keys, physical ID, physical keys, biometrics, etc.
- **Verifiers**: means by which assets can verify another asset's authenticator. Now covers password quality checks and verification, out of band and one-time key systems, physical locks biometrics, etc.
- **Access control**: a policy enforcement point preventing access by those lacking authorization.
- **Network controls**: includes firewall rules, address translation, reserved addresses, connection limiting, etc.
- **Vulnerability reduction**: software patching, software and pen testing, security certification, etc.
- **Host and data protection**: includes secure configuration and BIOS, data and message encryption, remote wiping for mobile devices, anti-malware, and physical protection of devices by their users.
- **User protection**: includes security training, spam filtering, etc.
- **Resource protection**: mainly used in connection with virtualised assets, includes redundancy and failover mechanisms, health monitoring, load balancing and elastic provisioning, etc.

RestAssured specific controls in the domain model for D7.2 are:

- **SecureEnclave**: applies to a Host device, means the host has hardware support for a secure enclave.
- **SecureExecution**: applies to a process, means the process is executed in a secure enclave, making it impossible even for another user with admin rights to access the process and its data.
- **StickyPolicy**: applies to data, an access control policy that defines what can be done with the data, linked to the data, and managed by a RestAssured Data Gatekeeper.
- **StickyPolicyEnforcement**: applies to processes, means there is an enforcement point, provided by a RestAssured Query Gateway, using a RestAssured Data Gatekeeper to decide if access is allowed.
- **StickyPolicyUI**: applies to a Human (i.e. a user role), means the user interface for that role includes mechanisms to specify the sticky policy for the user's data.
- **ParquetEncryption**: applies to a database, means the data is encrypted using columnar formats that allow efficient searching of the data.

There are also generic controls corresponding to some of these, added for D7.2. For example, homomorphic encryption was added, as a less efficient alternative to ParquetEncryption.

3.3.8 Adaptation as a security control mechanism

Since D7.2, new controls and control strategies were introduced into the knowledge base to represent the potential for system adaptation to be used as a means of addressing security requirements. Three types of adaptations have been found that could be used as security mechanisms to address risks:

- Where a threat could be addressed by a conventional security measure, but the asset where it is needed does not support the security measure, it may be possible to migrate to another asset that does.
- Where there is a threat to regulatory compliance linked to the jurisdictions in which certain parts of the system are operating, it may be possible to migrate parts of the system to different jurisdictions.

- If it is not possible to reduce risks by switching on a security measure or by migrating parts of the system, it may be possible to disable parts of the system to avoid the risk, albeit at the cost of reduced functionality.

Until D7.3, an SSM user performing a pre-deployment risk assessment could only reduce risks by specifying that security measures should be used to reduce the likelihood of a corresponding threat. Now, it is also possible to specify that a risk should be avoided through the use of adaptations.

The new controls used to model adaptations are:

- **Reconfigure**: means the asset retains it place and all its relationships, but its configuration is changed as part of an adaptation.
- **Move**: means the adaptation should break relationships of this asset with other asset(s), and recreate those relationships with a different asset. Typically applied to a Process asset.
- **Replace**: means the adaptation should break relationships of this asset with asset(s) that are being moved, and not recreate them. Typically applied to a Host asset, when moving a Process to a more suitable Host.
- **Merge**: if the assets at different nodes in the pattern are distinct, the adaptation should replace one with the other.
- **Split**: if the assets at different nodes in the pattern are the same asset, the adaptation should replace one with a distinct asset.
- **Disabled**: means the asset retains it place and all its relationships, but there is a configuration change (often at some other asset) preventing the disabled asset being used.

Security adaptations are modelled as control strategies. The first three adaptation controls indicate which assets would be subject to reconfiguration, migration or replacement. The remaining controls indicate the desired outcome.

Merge and Split are adaptation controls that describe desired outcomes involving a change in system topology. E.g. one might Merge two Jurisdiction assets to avoid regulatory compliance risks associated with a cross border data flow, or Split a Host so two hosted processes are no longer collocated so a DoS attack against one process that might overload its host does not compromise the other process.

Other security controls represent a persistent change in asset status rather than changes in system topology. E.g. one might wish to enable Secure Execution of a process using a Secure Enclave on its host, but if the current host doesn't provide access to a secure enclave these controls may be combined with replacing the host and moving the process.

Technically, the Disabled control is also a security control rather than an adaptation control. It must also become a persistent state of an associated asset in order to affect threat likelihood. E.g. one might disable a Data Flow, by a reconfiguration of the process that is the source of the data flow. Threats to the disabled data flow would then have the lowest possible likelihood, but only as long as the data flow remains disabled.

The main reason for adding these controls was to provide hints to the WP5 adaptation components to help narrow down the search for an adaptation response to a given threat. The idea is that when the adaptation components request a new risk level estimate from the RRE, it responds with a risk level, a list of threats giving rise to this risk level, and a set of adaptation control strategies that could be used to reduce the likelihood of one or more of these threats.

3.3.9 Inference rules

The main inference rules in the release corresponding to D7.2 are used to infer:

- the existence of Interface assets over which Hosts (devices) communicate with Logical Subnets to which they are locally connected;
- the existence of network paths between two Logical Subnets when there is a Host (a gateway device) connected to both Logical Subnets;
- the need for logical assets representing choices available to both sides when one Process (a Client) uses another Process (a Service);
- the presence of data flows when a process uses data, including flows through a chain of client-service connections where the ultimate user is not local to the data.

The last point is linked with the introduction of a wider range of relationships between processes and data, as discussed in Section 3.3.3.

There are also some 'domain specific reasoning' functions (encoded in SSM rather than in the domain model) to find extended paths from each logical subnet to each Host device, and communication channels between client and service process pairs.

3.3.10 Threat models

3.3.10.1 Threat overview

As noted in Section 3.2.6, threats in an SSM domain model are classified based on the circumstances in which they are relevant (expressed via a pattern of interacting assets), their causes and effects, and their possible control strategies. There is no simple relationship between threats in the domain model and most other classification schemes, because discriminants used in other schemes, such as the type of software vulnerability involved in a threat, are not of primary importance in an SSM domain model. Many different types of software vulnerability are mapped to the same threat class, since the nature of the vulnerability exploited is not considered important unless it affects the causes, effects or control options.

Because of this, the domain model was not created by mapping from some other knowledge base, although other knowledge bases are reviewed from time to time to see if anything has been missed. Instead, it was created by considering:

- what types of misbehaviours or changes in trustworthiness are relevant for each type of asset?
- what mechanisms could induce these, and how could those mechanisms be blocked or mitigated?
- which assets are necessary for each mechanism to function?

This led to the nomenclature currently used to denote threat classes, which comprises four parts, separated by dots:

- an initial character or two, indicating the type of domain model asset under threat, hence the H character at the start of the threat IDs in Fig 3.7 and 3.8;
- one to four characters indicating the type of effects, such as C, I or A representing confidentiality, integrity or availability breaches, or (for compliance threats) the breached regulations;
- a short string indicating the pattern of assets involved in the threat, such as RoH indicating the pattern related to a remote attack from a network on a host in Fig 3.7;
- a number used to distinguish threats, where the same effects can be achieved in the same situation but using different methods that require different countermeasures.

The threat notation does help ensure that no very significant threats (within the scope covered by the domain model) have been overlooked. A listing can easily be scanned to find asset classes not adequately covered.

Type of threat	Affecting	Notes
Internal software errors	Host or Process	Bugs that cause problems with no influence from an external agency. Variants cover different effects (crashes versus unreliable behaviour).
Physical intrusion	Space	The potential for attackers to enter spaces and gain physical access to other assets.
Physical tampering	Host	Attacks involving physical changes including theft and alteration of hardware or software.
Unauthorized local access	Host or Radio network	Attacks exploiting poor security against attackers who have local access, i.e. in a Space where the Host is located or Radio network can be accessed.
Unauthorized remote access	Host or Process	Models exploitation of an insecure configuration, e.g. failure to shut down unnecessary privileged services on a Host, or failure to authenticate at a service.
Local exploits	Host	Exploitation of software bugs to increase privileges from user rights to admin rights.
Remote exploits	Host or Process	Exploitation of software bugs to cause a crash or acquire user or admin rights over a device, control a service or access a back end database, or to control or impersonate a client process. Attacks on clients typically involve a service, e.g. using XSS attacks, often using phishing as a delivery vector.
Malicious attachments	Email client	Malicious email attachments designed to control the email client. Combined with other (local) exploits covering subsequent actions, e.g. privilege escalation or confidential data leakage.
Malware attacks	Host (or Data)	The introduction of malware into a device. Variants cover different ways this could be done, including malicious attachment, phishing, local access using removable storage media, and worm propagation. Ransomware attacks on data assets are also modelled as a special case.
Snooping attacks	Comms Channels	Variants cover the means by which an attacker can access messages, either on a radio network or at a host through which messages pass.
Spoofing attacks	Radio networks, Processes	Covers impersonation of a radio network by an attacker with access to the space in which the legitimate network operates, or of a client or service to the other exploiting failures of authentication or subversion of network routing functions.
DoS attacks	Interface or Service	Overloading a target IP address (i.e. an interface between a host and a local subnet) or a service by sending excessive or malicious messages, consuming bandwidth or other resources needed for communication.

Type of threat	Affecting	Notes
Exploitation of privileges	Various	These threats model legitimate access rights and are used to create models of attack paths in a system. They are also used in models of types of attacks that cannot easily be captured as a single threat. For example, most snooping attack models involve a step to gain access to messages, and the snooping itself is then an exploitation of privileges gained. In most cases, there are no control strategies because privileges can't be blocked without harm to legitimate users. In a few cases this is not true, e.g. one could use trusted hardware to protect processes and data from the administrator of their host.
Secondary threats	Various	These threats model effect propagation mechanisms and are used to create models of secondary effect cascades. They cover propagation of overload and its effect on availability, propagation of availability issues, propagation authenticity and integrity issues via data flow dependencies, the effect of leaky communication channels, and causes and effects of unreliability.

Table 3.8: Threat Coverage

Threat models for remote attacks on hosts assume the attacker sends their malicious messages on the local subnet to which the host is connected. This can be combined with the threat of attackers sending messages between subnets, as discussed in Section 3.2.6.6. Such attacks can be blocked by introducing firewall rules to prevent the transmission of messages to the target host.

Threat models for remote attacks on services, including attacks on hosts via a hosted service, cannot be blocked by firewall restrictions if the attacker has access to a subnet on the path from a legitimate client. This is because even if one introduces a rule to block messages, exceptions will need to be added allowing messages from the client to reach the service. Threat patterns therefore include the presence of a client process and finds subnets on the path from this client. Attacks on or via services from other subnets are not modelled separately, since they can be blocked using firewall restrictions and so are considered equivalent to direct attacks on the host.

3.3.10.2 OWASP Top 10 coverage

One piece of feedback from OCC on the domain model from D7.1, was that coverage of the OWASP Top 10 threats [37] was important and should be made more transparent. As discussed in Section 3.2.4.2 of D7.1 [33], many OWASP Top 10 threats were covered but because there is no direct mapping between SSM threat models and attack mechanisms (except where they require different control strategies) the correspondence was unclear. For OCC this was a problem, because most of their products are web applications, and their customers have heard of the OWASP Top 10 and want evidence that the software is not vulnerable.

To address this, some threats were added in D7.2 to fill gaps in the OWASP Top 10 coverage, notably for attacks involving client-side interactions. The threat descriptions (which appear in the SSM GUI and reports) were then modified to include references to OWASP Top 10 threats.

OWASP Top 10 Threat (2017)	**Coverage in D7.2 domain model**
A1: Injection of hostile data to an interpreter	DB.MTW.PRoSDB.1: covers query injection via a service to a DB process. A DB process is one that uses a well-known command language, so this covers injection into SQL and other languages such as LDAP, etc.
A2: Broken authentication	SC.Auth.PRoS.2: covers impersonation of a legitimate client using credential stuffing to discover their authentication password. SC.Auth.PRoS.3: covers impersonation of a legitimate client when there is a failure or absence of any authentication mechanism.
A3: Sensitive data exposure	D.C.HD.1: covers access to data having gained control over the host where it is stored as clear text. D.C.PrD.1: covers access to data by taking control of a process (e.g. gaining access to data in a browser). D.C.SCDFSC.1: covers leaking of data to an imposter client. D.C.CDBPath.1: covers leaking of data to an imposter service. D.C.DBCChannel.2: covers snooping messages from a service to a client. D.C.CDBChannel.2: covers snooping messages from a client to a service.
A4: XML External Entities	P.MTW.PRoS.1: covers remote exploitation of a vulnerability to control a service process. The vulnerability may be a failure to sanitize untrusted XXE references, but it could also something else related to OWASP 2017:A9.
A5: Broken access control	SC.Auth.PRoS.1: covers a failure to control access, allowing an attacker to use a service as though they were a legitimate client.
A6: Security misconfiguration	H.MTW.PRoS.1: covers injection of host commands via a vulnerable service. H.MTW.RoH.2: covers exploitation of an insecure configuration (e.g. failure to change default passwords or eliminate unnecessary services) to control a host.
A7: Cross-site scripting	SC.Auth.LRXSS.1: covers reflective XSS attack using phishing to send a malicious URL via email exploiting a vulnerability in a trusted service. SC.Auth.WRXSS.1: covers reflective XSS attack using phishing to send a malicious URL via webmail exploiting a vulnerability in a trusted service. SC.Auth.SXSS.1: covers a stored XSS attack, in which the malicious URL is inserted on the service side, e.g. in a social network, and not sanitized by the service before being served to another client.
A8: Insecure deserialisation	This aspect is not yet explicitly covered in D7.2. Some threats including SC.Auth.SXSS.1 are related to A8, but references to A8 are not yet included in threat descriptions.
A9: Using components with known vulnerabilities	H.A.RoH.1: covers remote exploitation of a vulnerability to crash a host. P.A.PRoS.1: covers remote exploitation of a vulnerability to crash a service. H.MTW.RoH.1: covers remote exploitation of a vulnerability control a host.

OWASP Top 10 Threat (2017)	Coverage in D7.2 domain model
	P.MTW.PRoS.1: covers remote exploitation of a vulnerability to control a service process.
A10: Insufficient logging and monitoring	This aspect is not yet explicitly covered in D7.2.

Table 3.9: OWASP Top 10 Threats

Monitoring controls have been introduced against host and service availability threats, but not yet to detect other types of breaches. Logging controls are also used for compliance purposes under the GDPR, but this focuses on logging processing actions, and not asset misbehaviour.

3.3.10.3 GDPR compliance threats

The current models of GDPR requirements cover GDPR Articles 6, 8 and 9:

Article 6 defines what constitutes an acceptable legal basis for data processing, including by consent of the data subject.

Article 8 covers additional measures needed when processing data from children.

Article 9 defines special categories of data requiring additional protection, and a subset of categories for which member states may impose additional regulations going beyond the GDPR.

The initial models of these requirements were developed in SHiELD, and were recently refactored in SHiELD. Some updates were subsequently made in RestAssured for consistency with the RestAssured models of data flow, and to add control strategies that use RestAssured security technologies from WP4 and WP6.

Threat models from SHiELD covered:

- D.GDPR.HDHu-J.1 and D.GDPR.HPDHu-J.1: cover failure to define the jurisdiction for a host that stores or processes data: the GDPR does not require this, but unless the jurisdiction is defined, cross-border data flows cannot be detected;
- P.GDPR.HuHPrD.1 and P.GDPR.HuCSD.1: cover the need for a legal basis as defined by Article 6 when data is processed either locally or remotely;
- P.GDPR.HuPD.1: covers the need for additional measures specified by Article 8 when processing data from children;
- P.GDPR.HuHPrSD.1 and P.GDPR.HuCSSD.1: cover the need to use additional measures for special categories of data as defined by Article 9 when the data is processed locally or remotely.

Threats P.GDPR.HuHPrD.1 and P.GDPR.HuCSD.1 have control strategies, each of which covers one of the circumstances under whihc processing of personal data is lawful, as defined in GDPR Article 6.1, whose subclauses are:

a) the data subject has given consent to the processing of their personal data for one or more specific purposes;

b) processing is necessary for the performance of a contract to which the data subject is a party, or to take steps at the request of the data subject prior to entering into a contract;

c) processing is is necessary for compliance with a legal obligation to which the controller is subject;

d) processing is necessary in order to protect the vital interests of the data subject or of another natural person;

e) processing is necessary for the performance of a task carried out in the public interest or in the exercise of official authority vested in the controller;

f) processing is necessary for the purposes of the legitimate interests pursued by the controller or by a third party, except where such interests are overridden by the interests or fundamental rights and freedoms of the data subject which require protection of personal data, in particular where the data subject is a child.

Most of these involve non-technical measures, but the first requires certain technical measures to be implemented, and all require an auditable record to be created. RestAssured provides technologies that can be used to implement consent-based control over processing, so the control strategies from SHiELD corresponding to Article 6.1(a) were replaced by strategies based on the use of RestAssured sticky policies as developed in WP6. The same is true for threats P.GDPR.HuHPrSD.1 and P.GDPR.HuCSSD.1 which cover additional measures needed for special categories of data. These control strategies are covered separately in Section 3.3.10.4.

In SHiELD, the focus is on protecting health data, which is one of the special categories of data defined in Article 9. In RestAssured we are not concerned only with health data, but also other types of data. In RestAssured, the asset class hierarchy was therefore expanded to include subclasses of Data assets that are subject to special treatment under Article 9. This is reflected in the asset class hierarchy from Figure 3.12. A subclass SensitiveData is used to represent data that falls within any of the special categories defined by Article 9. Further subclasses represent genetic, biometric and health data subcategories that are also subject to national regulations under Article 9.4. These Data subclasses are used in the RestAssured versions of GDPR compliance threats.

RestAssured also added one new GDPR compliance threat, called DF.GDPR.NRDHuDF.1, shown in Figure 3.13. This relates to data flows across borders within the EU when the data is one of the categories defined by Article 9.4 as being subject to additional (national) regulations, namely genetic data, biometric data or data concerning health. Further asset subclasses were introduced to represent these categories, and are used in the threat model for DF.GDPR.NRDHuDF.1.

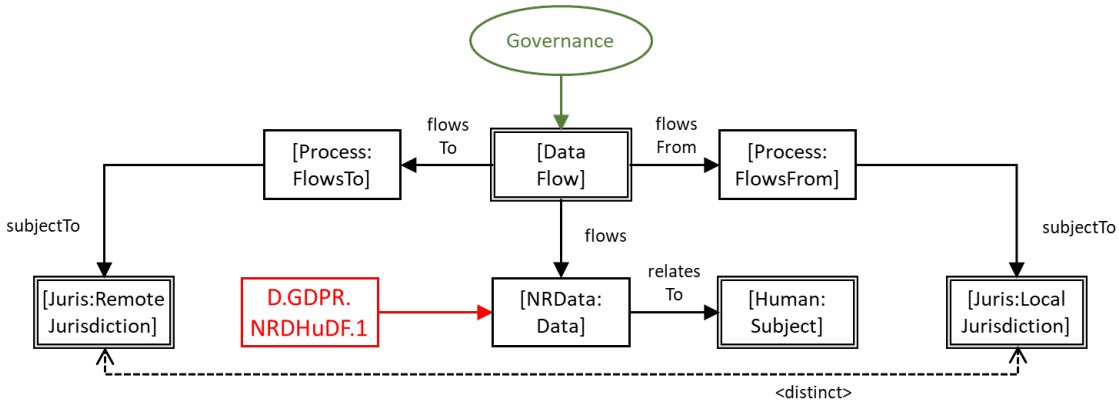

Figure 3.13: Risk of illegal cross-border transfer of nationally regulated personal data

This situation is covered by SHiELD but only for cross-border exchanges of health data. Health data is one of the categories subject to national regulations under GDPR Article 9.4, so agreements between some member states have created a trust network of so-called National Control Points (NCPs). Cross-border

transfer of health data is permitted via the NCPs, and the SHiELD threat model captures the need to use that mechanism. The SHiELD equivalent of DF.GDPR.NRDHuDF.1 is therefore too limited (only considering cross border flows of health data), and the control strategy too specialised to be used in RestAssured.

In the RestAssured PAYD scenario, the issue relates to the use of biometric measurements from a driver, as biometric data is also one of the categories subject to national regulation under GDPR Article 9.4. If the driver crosses a border, and there is no other change in the system, their data will be transferred across that border to a cloud service in their country of origin for storage and analysis.

The control strategy in Figure 3.13 is simply that the data flow is subjected to a 'governance' control. This is a generic control representing non-technical measures based on analysis leading to a decision that the threat is not an issue. In this situation one might analyse legislation on either side of a border and determine that it is OK to transfer the biometric data in that case. However, one could also address the threat by an adaptation that turns off the data flow for the biometric data (which continuing to stream other vehicle data), or by migrating the service that analyses the data to a data centre in the same jurisdiction as the vehicle. These adaptations are not control strategies because they do not make the pattern in Figure 3.13 acceptable. In each case the adaptation breaks the pattern so the threat is no longer present.

3.3.10.4 Threats countered by RestAssured security measures

There are two sets of controls corresponding to RestAssured security measures:

- controls involving the use of advanced encryption and secure hardware, developed in WP4;
- controls involving the use of sticky policies associated with data, developed in WP6

The main use of sticky policies in RestAssured is to implement the means to acquire consent from the data owner for processing their data. This includes processing personal data by consent of the data subject, leading to a RestAssured control strategy for some of the threats taken from SHiELD.

Where the data is processed locally (on the same Host device where it is stored), the threat from SHiELD is P.GDPR.HuHPrD.1, which is one of the examples shown in Section 3.2.6.7 in Figure 3.11. The RestAssured control strategy using sticky policy components is shown in Figure 3.14:

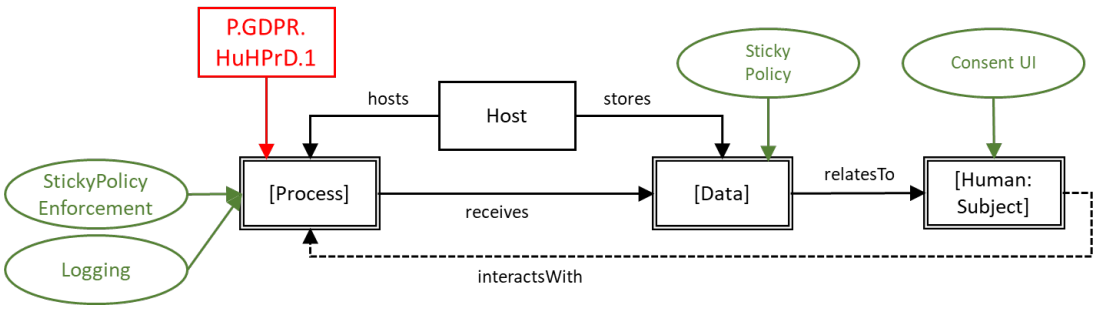

Figure 3.14: Implementation of local processing by consent using RestAssured sticky policies

Here, the sticky policy enforcement control represents the Process obtaining an access token from a RestAssured Data Gatekeeper, which decides whether to grant access based on the sticky policy attached to the data. The consent UI control associated with the data subject signifies that the user interface for that user role must provide the means for them to grant or withdraw consent. A logging control captures the GDPR requirement to maintain records of the processing so it is possible to tell the subject who has processed their data should they request that information, and also to prove that a valid consent was in force at the time.

If the data is processed remotely, the pattern of assets and the positioning of technical security measures is slightly different, as shown in Figure 3.15.

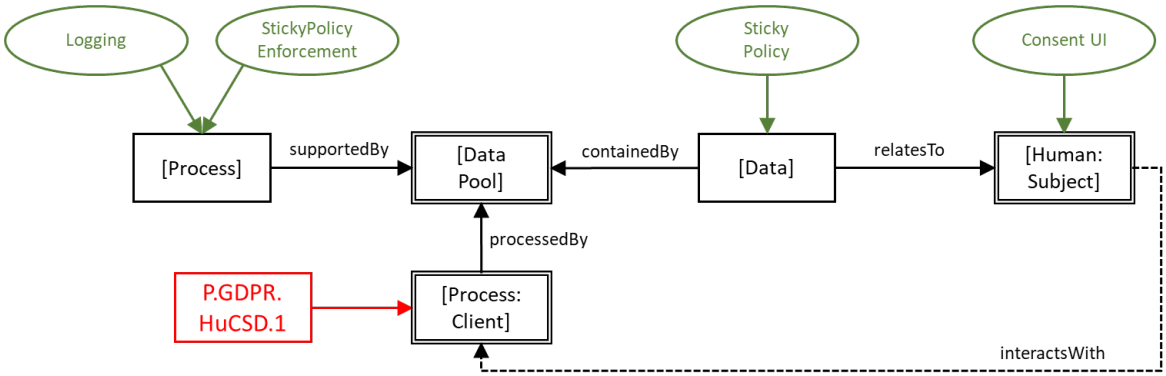

Figure 3.15: Implementation of remote processing by consent using RestAssured sticky policies

Here the processing is handled by the Client process, which is why that is the process not interacting with the data subject (if it were, the data subject would be controlling the process themselves and the risk would not be applicable). However, the client process could be something as simple as a web browser, so it is not sensible to insert an enforcement point or logging there. The data is obtained via a second data serving process, and that is the appropriate position in the system for these controls.

The same control strategies apply also to the case of sensitive data, except the consent UI must then obtain informed consent, meaning the data subject must be informed about risks as well as benefits. None of the RestAssured scenarios are explicitly dealing with special categories of data defined by Article 9, except the use of biometric measurements in the PAYD scenario as discussed in Section 3.3.10.3.

Advanced encryption controls developed in the first half of the project in WP4 focused on supporting the use of secure hardware enclaves to protect data while being processed. This addresses threat P.MTW.HP.1, representing the ability of someone with control of a host using admin rights to take control of a process running on that host.

The reason this threat is of interest in RestAssured is that in cloud applications, the cloud service provider has admin privileges which they may abuse, and other tenants may be able to acquire admin privileges if the cloud infrastructure fails to isolate virtual devices belonging to each tenant. Although the immediate threat is to a process, it provides a possible attack path to gain access to data used by the process. It is possible to protect data stored in the cloud by encrypting the data prior to uploading it to a virtual server. But if the data is processed in the cloud, then the process doing this must be sent a decryption key, so the data is vulnerable while it is being processed.

Secure enclaves provide a solution through a protected memory space into which even someone with root privileges cannot peek, and an embedded security key allowing users to establish a chain of trust for the deployment of processes and provisioning of keys needed to access the data. RestAssured WP4 developed a framework supporting this, allowing a process to be executed inside the secure enclave, given that the host device has the necessary features. The control strategy is shown in Figure 3.16.

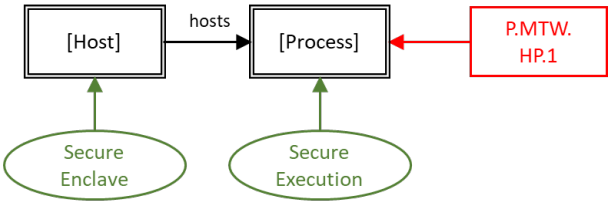

Figure 3.16: Protecting data during processing via RestAssured support for secure enclaves

In the second half of the project, WP4 increasingly focused on protection of stored data using Parquet encryption. This exploits an efficient storage schema for big data searches, and the encryption is applied in

such a way that processes only access the fields they need.

The domain model from D7.1 already has a control strategy modelling the use of encryption to protect data being stored. The novelty of Parquet encryption is that it imposes a low overhead for query processing, so from a security perspective there is no need for a new control strategy. The existing threat and control strategy models are enough to capture the threats to data, and the use of encryption to intervene in relevant attack paths. They show that Parquet encryption used in conjunction with secure enclaves and sticky policies provides a safe way to perform data analytics efficiently in the cloud.

3.3.11 Modelling error checks

Compliance threats and compliance sets were introduced in Section 3.2.6.7. A compliance set represents a set of requirements that a system should meet. Each requirement is modelled as a compliance threat, which has no causes and effects but captures situations that breach a requirement, or in which controls are needed to meet the requirement.

The RestAssured domain model corresponding to D7.2 contains a GDPR compliance set representing a subset of requirements imposed by the GDPR, as discussed in Section 3.3.10.3. However, the requirements modelled by a compliance set are not always regulatory requirements. Compliance sets can also be used to model best practice recommendations, and in D7.2 one was introduced to capture possible SSM system modelling errors. This was done mainly in response to observations from validation experiments and user workshops with OCC and Adaptant, in which it was seen that users may overlook features of the system.

Modelling errors of this type should be regarded as threats because overlooking a system feature may lead to other threats being missed by the automated threat identification procedure supported by SSM. In D7.1 this was addressed by the use of validation patterns, which are similar to construction pattern, except that the missing element of the model (an asset or link) cannot be inferred. For example, every process must run on a host, so D7.1 contained a validation pattern (still used in D7.2) that detects the presence of processes with no host. This cannot be a construction pattern because it is impossible to infer the necessary fix, which may be to add a missing host, or only to create a link between the process and an existing host. Validation patterns cause SSM to display the inbound or outbound relationship panel for an asset, and prompt the user to add a link, leaving them to decide whether this is with an existing asset or one they must first add.

Validation patterns are useful in such simple situations, but it became apparent that the types of errors made by OCC and Adaptant were far less simple. The most common errors we noticed were:

- failure to specify interactions between intermediate processes and data, so some data flows would not be constructed, and associated threats of cross-border transfer and propagation of integrity issues would be missed; and
- specifying interactions between processes when there was no network path linking their hosts, so the data flow would have no associated communication channel, and snooping and spoofing threats would be missed.

The first error was partly addressed by using more sophisticated, iterative construction rules to generate data flows. However, it is still possible for a data flow to be omitted entirely if either the data source or consumer is not specified. To address this, three modelling error threats was added. The first of these is P.E.PsDdH.1, shown in Figure 3.17, the character E after the first dot signifying this is a possible modelling error.

This threat encodes the fact that when a process serves data, it is normally collocated on the same host as the data. It is possible for a process to serve data obtained from another process, but this is now implied by an iteratively constructed data flow asset. The user should therefore specify a single data serving process running on the host where the data is stored, and if they don't it is usually because the wrong process was linked to the data, or the wrong host was linked to the serving process.

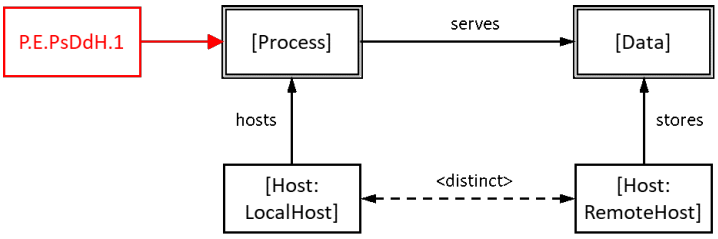

Figure 3.17: Detecting servers of non-local data

Threat DP.E.DPnP.1 shown in Figure 3.18 detects cases where data is being served (using a constructed Data Pool asset to detect this is the case), yet no remote process is using that data. In previous validation experiments, this most commonly happened when the relationship between a consumer process and the data had been overlooked.

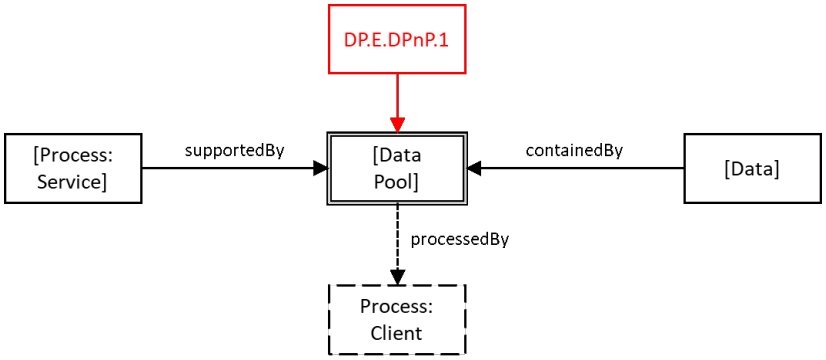

Figure 3.18: Detecting data that is served but not used

Another common error is to specify the link with a remote consumer process, but overlook the fact that a local process must also serve the data where it is stored. Threat C.E.DHCHdH.1 shown in Figure 3.19 detects this type of error, which may also occur when the wrong host is specified as the one storing the data.

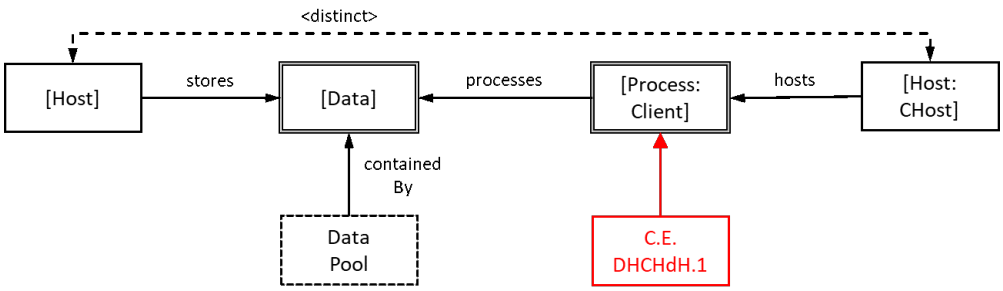

Figure 3.19: Detecting data that is used but not served

For completeness, one further threat was added, covering the case where the data serving process and the data consumer are both specified correctly, but there is a break in the chain of intermediate client-service interactions by which the data gets between the two. The threat is then that the data flow is not constructed, and associated threats would be missed. This is covered by threat D.E.CDUnoDP.1 shown in Figure 3.20.

Finally, we come to threats C.E.CSSPHHnSC.1 and C.E.CSSPHHnSCS.1 shown in Figures 3.21 and 3.22. These cover the case where there is a client-service interaction (and an associated data flow is generated), but there is no network path between the two hosts (so snooping and spoofing threats would be missed). The threat patterns detect the absence of a constructed communication channel between a client-service pair.

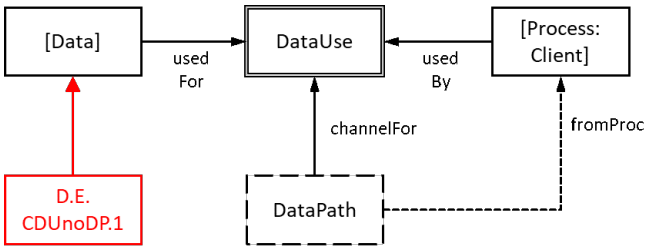

Figure 3.20: Detecting data that is used but not served

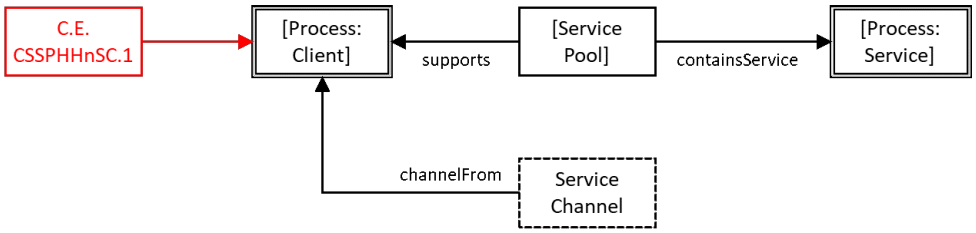

Figure 3.21: Detecting data that is used but not served

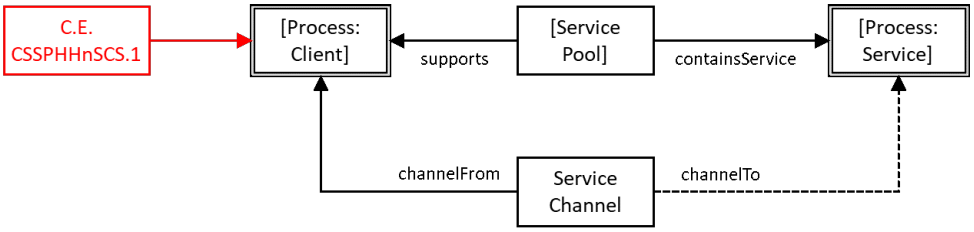

Figure 3.22: Detecting data that is used but not served

There are similar threats referring to end-to-end data flows between data server and consumer processes, but these are not shown here as they are just larger patterns in which those from Figures 3.21 and 3.22 are embedded. The reason they were added to the compliance set is because sometimes when there is a disconnected client and service, the problem is that the wrong intermediate processes have been chosen, rather than the lack of a communication path between those that were chosen. Although the threats detect the same situation, they refer to a different context and possible solution.

3.4 Mapping from ReAs-CSAP to Secure System Modeller

This section describes a mapping from the different elements of the ReAs-Cloud System Analysis Pattern (ReAs-CSAP) to the corresponding constructs of the System Security Modeller (SSM). This mapping is outlined in Table 3.10. Furthermore, the different ReAs-CSAP- and/or SSM-elements are assigned to the corresponding risk assessment concepts that are represented by the different elements. The mapping represents an update of the mapping in Deliverable 7.1 (see [43], Section 3.3).

- *Goals* consider the objectives and restrictions regarding the desired and/or required level for the security and privacy of assets. These objectives can be, for example, specified by laws, regulations or contracts.
- *Assets* represent any item that has value to an organisation or a person. Assets can be classified into *primary assets* and *supporting assets*. Primary assets include business processes as well as information and data. Supporting assets support the provision of business processes. In this context, they

can be responsible for processing, storing and transferring the relevant data. Examples for supporting assets are software, hardware or network components.

A risk analysis is performed regarding to the security and privacy properties of assets. Here, the value of an asset is assessed. In this assessment the impact for the case that security or privacy properties of an asset get compromised is considered. Furthermore, the likelihood for such security or privacy incidents are assessed. Based of the asset value and the likelihood for a security or privacy incident the risk level for an asset is determined.

- *Threats* are potential events that can lead to a loss of the security and privacy properties of assets. In the context of a risk analysis, the likelihood for a actual occurrence of a threat is assessed. Examples for threats are attacks of hackers, system failures or incorrect configurations of cloud components.
- *Mechanisms* are countermeasures for supporting the implementation of the goals regarding the security and privacy properties of assets. Here, mechanisms shall reduce the likelihood that relevant threats lead to security or privacy incidents.

Table 3.10: Mapping from ReAs-CSAP-elements to System Security Modeller-elements

Risk Assessment Concept	ReAs-CSAP-element	ReAs-CSAP-instance type	SSM-element	logical equality
Goal	Indirect Stakeholder	Legislator	Jurisdiction	yes
	Indirect Stakeholder	Contract	Impact Level	no
	Indirect Stakeholder	Domain	Impact Level	no
	Indirect Stakeholder	Assessor	Stakeholder: Adult, Stakeholder: Organisation	yes
Asset	Cloud Element	Service	HostedAsset: ApplicationProcess	no
	Cloud Element	Pool	Space: DataCentre; implies the presence of infrastructure resources	no
	Cloud Element	IaaS	HostedAsset: ApplicationProcess	no
	Cloud Element	PaaS	HostedAsset: ApplicationProcess	no
	Cloud Element	SaaS	HostedAsset: ApplicationProcess	no
	Cloud Element	Resource	HostedAssets, NetworkAssets	partially
	Cloud Element	Hardware	NetworkAssets	partially
	Cloud Element	Software	HostedAsset: ApplicationProcess	partially
	Cloud Element	Location	Space: DataCentre	yes
	Cloud Element	Cloud Software Stack	HostedAsset: ApplicationProcess	no
	Cloud Element	Development Env. / API	HostedAsset: ApplicationProcess	-
	Cloud Element	Software Product	HostedAsset: ApplicationProcess	no
	Cloud Element	Data	HostedAsset: Data, HostedAsset: SensitiveData	yes
	Cloud Element	Sticky Policy	Control	no
Actor	Direct Stakeholder	Data Subject	Stakeholder: Adult, Stakeholder: Organisation	different levels of detail
	Direct Stakeholder	Data Controller	Stakeholder: Organisation	different levels of detail
	Direct Stakeholder	Cloud Provider	Stakeholder: Organisation	different levels of detail
	Direct Stakeholder	External RestAssured Provider	Stakeholder: Organisation	different levels of detail

Table 3.10: Mapping from ReAs-CSAP-elements to System Security Modeller-elements (cont.)

Risk Assessment Concept	ReAs-CSAP-element	ReAs-CSAP-instance type	SSM-element	logical equality
	Direct Stakeholder	Data Consumer	Stakeholder: Adult, Stakeholder: Organisation	different levels of detail
	Direct Stakeholder	Online Service Developer	Stakeholder: Adult, Stakeholder: Organisation	different levels of detail
	Direct Stakeholder	Cloud Administrator	Stakeholder: Adult, Stakeholder: Organisation	different levels of detail
	Direct Stakeholder	IaaS Operator	Stakeholder: Organisation	different levels of detail
	Direct Stakeholder	SaaS Operator	Stakeholder: Organisation	different levels of detail
	Direct Stakeholder	Cloud Support	Stakeholder: Adult, Stakeholder: Organisation	different levels of detail
Mechanism	Cloud Element	RestAssured Platform	Control	partially
	Cloud Element	Data Protection Component	Control	partially
Threat	none	none	Threat	-

4 Tools

4.1 CSAP-tool

We provide tool support for our pattern, namely the CSAP-tool. This tool provides a graphical editor that supports the creation and instantiation of the ReAs-CSAP. To this end, the tool provides two editors, namely the *Designer-editor* and the *User-editor* (see Figure 4.1). These editors are explained shortly in the following sections.

A detailed description of the CSAP-tool functionality has already been given in Deliverable 7.1 (Section 4.1).

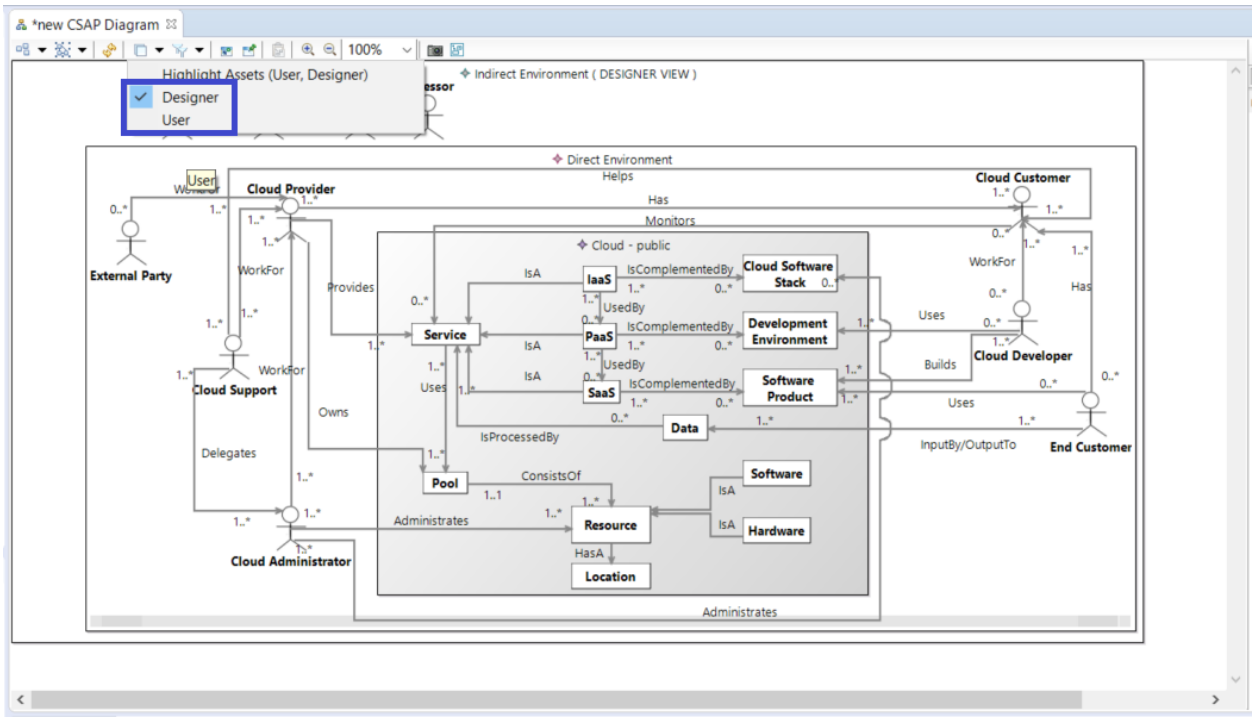

Figure 4.1: Editor modes of the CSAP-tool

4.1.1 The Designer-editor

The *Designer-editor* of the CSAP-tool enables designers to create domain-specific Cloud System Analysis Patterns. For example, the ReAs-CSAP (see Figure 3.1) represents a specific CSAP within the domain of the RestAssured project. Designers can create their own cloud patterns by adding *Indirect Stakeholders*, *Direct Stakeholders* and *Cloud Elements* to the CSAP and/or modifying existing CSAP-elements. Furthermore, associations between Direct Stakeholders and Cloud Elements as well as between Direct Stakeholders among each other can be created and/or modified. The generic approach of the Designer-editor allows a definition of different types of Direct Stakeholders, Indirect Stakeholders and Cloud Elements in a specific CSAP. While adding a CSAP-element, its name can be defined. In the Designer-editor this name of a CSAP-element defines also its instance type.

Figure 4.2 shows an example for the allocation of a new logical meaning to a Cloud Element by changing its type. Because the ReAs-CSAP shall also allow the representation of Data Protection Components that are not provided directly by the RestAssured platform, the type of the Cloud Element *RestAssured Platform* is changed to *Data Protection Platform*. Figure 4.3 shows the properties that can be defined for the Cloud

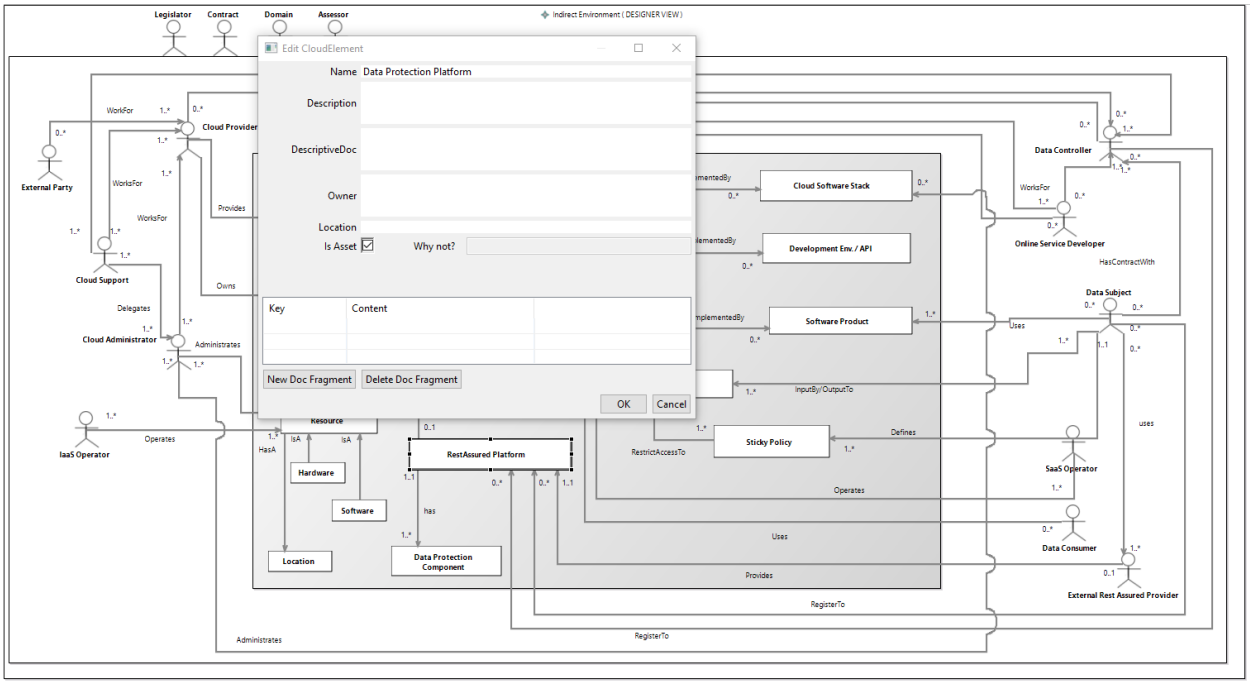

Figure 4.2: Modification of a Cloud Element in the ReAs-CSAP

Element *Data Protection Platform*.

For a more refined version of the ReAs-CSAP, the Cloud Element *Data Protection Components* could by replaced by Cloud Elements that represent certain components of the RestAssured platform, e.g. the *Data Gatekeeper* or the *Risk Assessment component*.

4.1.2 The User-editor

The *User-editor* enables the instantiation of any CSAP. In the context of the RestAssured project, the User-editor is used for the instantiation of the ReAs-CSAP regarding the different use cases within the project. A detailed description of the ReAs-CSAP instances for the use cases is given in Deliverable 7.1 (see [43], Section 3.1.2.4 to Section 3.1.2.6).

As an example, Figure 4.4 shows the definition of the type of cloud during the instantiation of the ReAs-

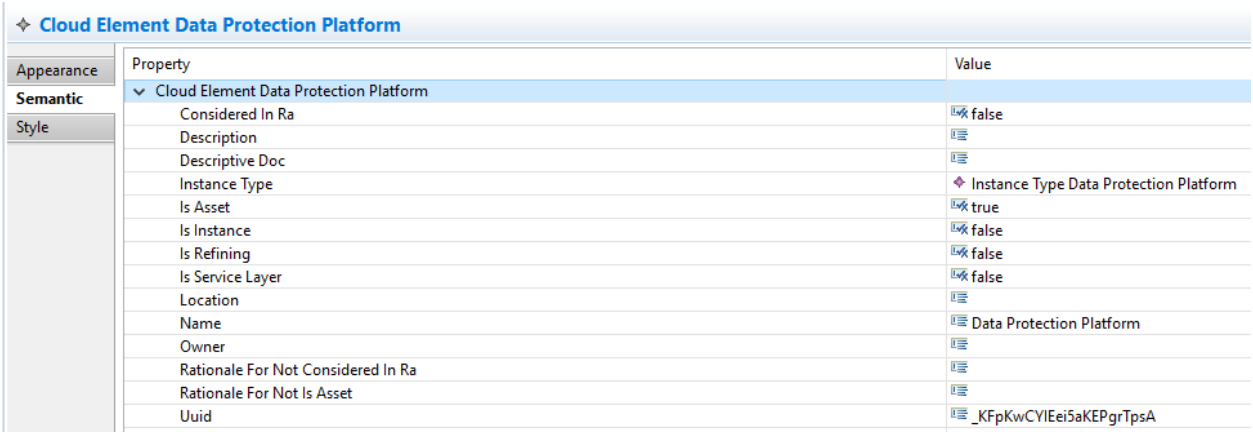

Figure 4.3: Properties of the Cloud Element of the type Data Protection Platform

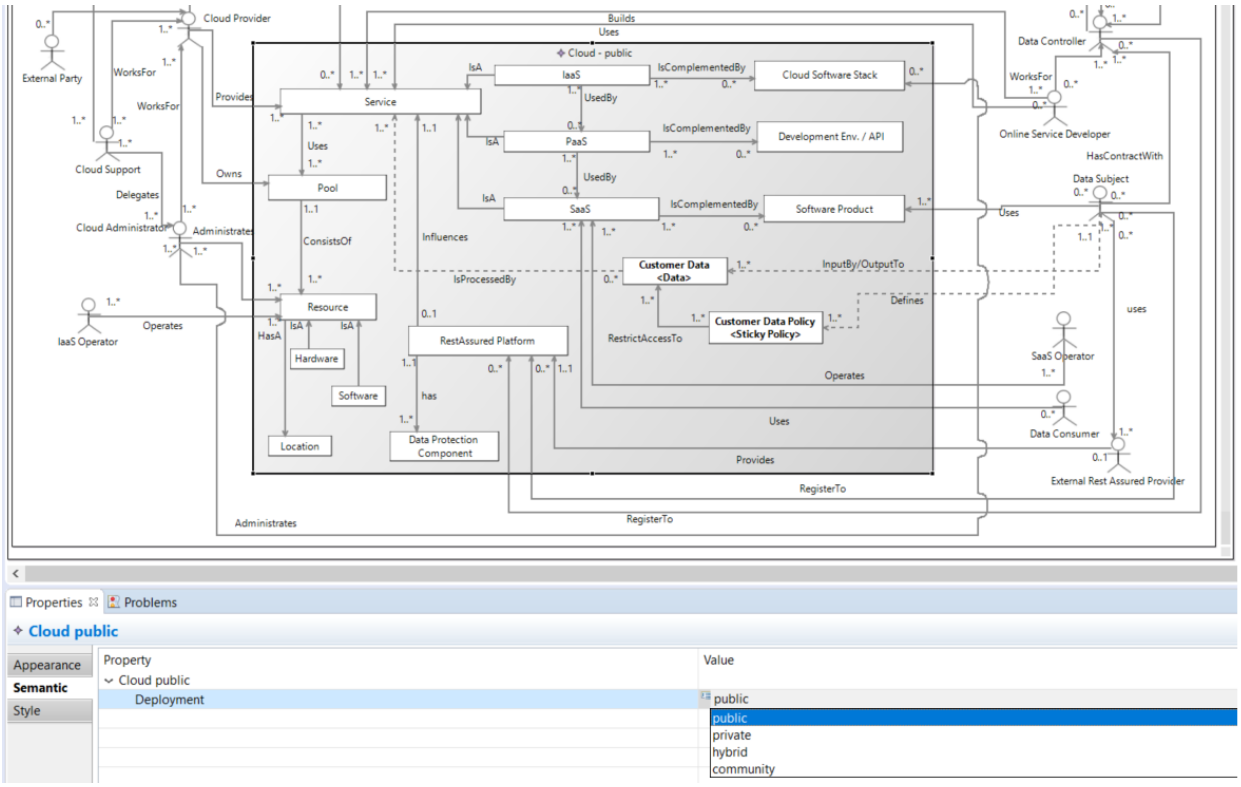

Figure 4.4: Specification of the type of the Cloud

CSAP. This information is especially relevant for the consideration of the deployment.

4.2 System Security Modeller

The System Security Modeller (SSM) is a web-based graphical tool which allows a system designer or analyst to perform a design-time risk assessment of a system. This section describes the tool, and the extensions for RestAssured.

4.2.1 Architecture

The System Security Modeller UI is a web-based user interface for a system of back-end components. Figure 4.5 gives an overview of the high-level architecture of the constituent services and underlying triple store. The services are accessed through a REST API, either using the System Modeller UI, or some other application. The interface exposes selected parts of the models, such as collections of assets or threats (read only) or update methods to allow users to edit models. These calls are sanitised and passed to the underlying querier and validator components. A low-level semantic store component allows access to the actual triple store.

As shown in Figure 4.5, the store contains one core model, and potentially multiple domain and system models (Section 3.2.1). All models are stored in separate graphs, allowing for easy import and export.

4.2.2 User Interface

Figure 4.6 shows the canvas of System Security Modeller with a model just created. The process for using SSM involves the following steps:

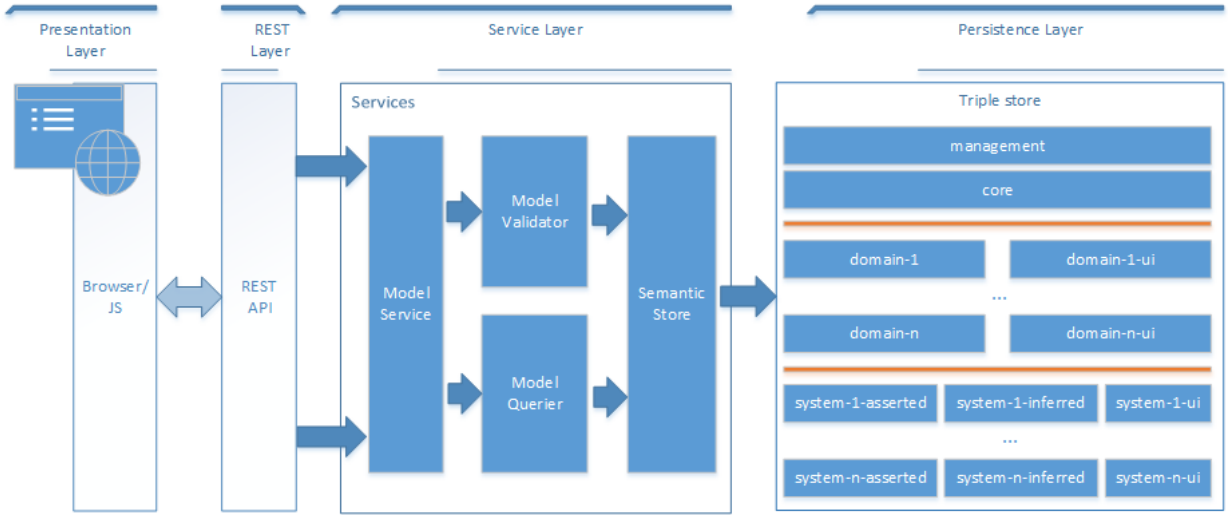

Figure 4.5: The System Modeller high-level architecture

- add assets to the model, giving them meaningful labels, and setting cardinalities (some assets by design are singletons e.g. the internet)
- connect the assets using the allowed relationship types from the domain model (depends on the types of the assets being connected)
- validate the model to generate inferred assets (these are implicitly defined through the relationships between other assets)
- set trustworthiness levels for asset attributes, and impact levels for misbehaviours
- calculate risk levels to determine which threats pose the highest risk

Figure 4.6 shows a very simple example system model representing an online shop as seen by the shop's operator. The customer's home is on the left, using a PC connected to the Internet (the details of the customer's home network are omitted because the shop's operator has no control over that). On the right are the main assets of the shop: a website with a back-end database that contains personal data about the client, each running on a server connected to the DMZ and LAN subnets respectively. A router/firewall provides connections between these subnets and the Internet, providing the network path by which the client can access the website.

Once the threat catalogue has been generated and risk levels evaluated, one can view the complete threat catalogue in the right-hand panel, as shown in Figure 4.7. This shows threats to the asset selected in the canvas, but if no asset is selected (which is the case in this screenshot), all system threats are shown.

The most useful ordering function arranges threats in order of decreasing risk level, making it easy to find those that most need to be addressed. Even so, there may be a lot of threats at an equally high level, especially if few security controls have been selected in the model. We recommend adding basic security first, e.g. by following a check list such as the UK Cyber Essentials scheme which advocates the use of firewalls, secure configuration of devices including setting passwords and disabling guest accounts, imposing access controls, using anti-malware tools, and applying security patches promptly in a systematic way. Once some security is included in the model, the number of high-risk threats will be smaller, and it will be possible to find gaps that require specific security treatment. In Figure 4.7, these basic security measures have been applied, but five very-high risk threats still require treatment.

It is often helpful to add a filter to the threat list, to exclude secondary threats (for which controls are rarely available), or to focus only on root causes – i.e. the original disruption at the start of the attack path

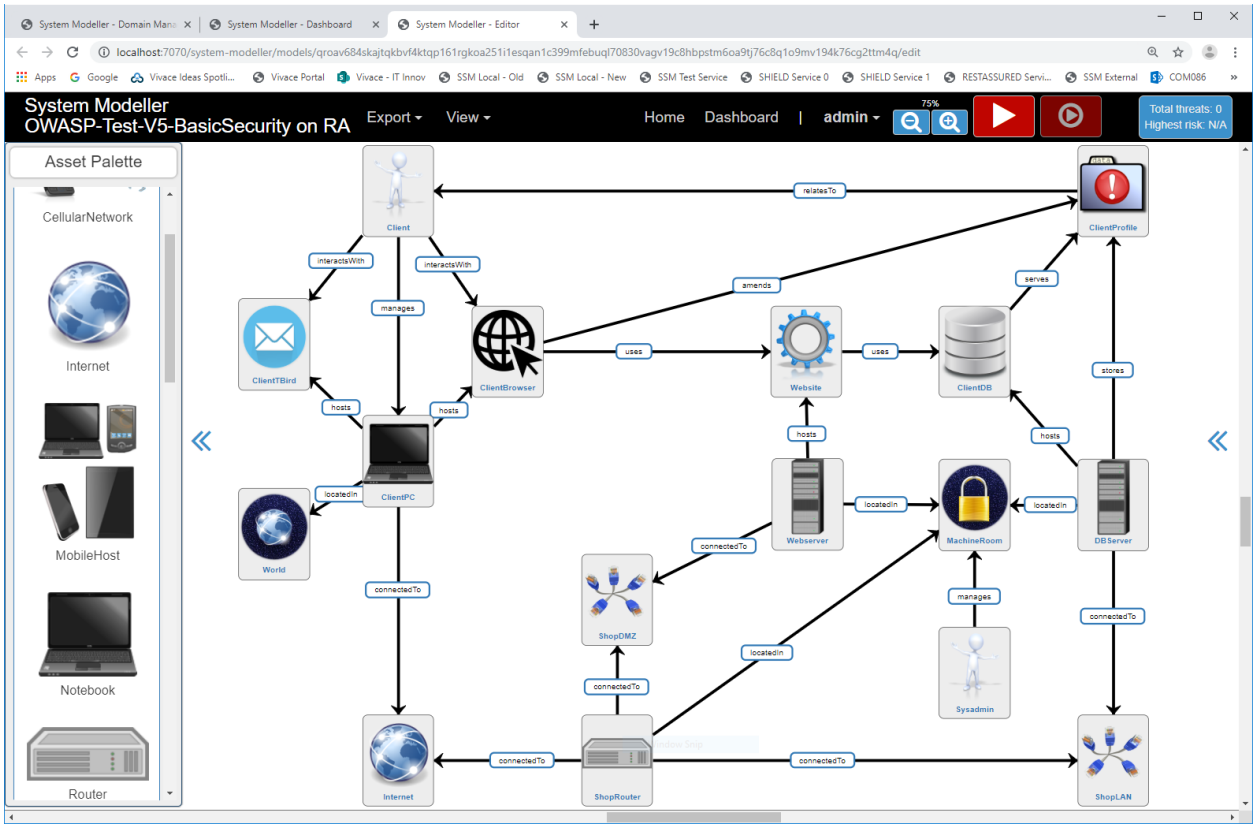

Figure 4.6: The SSM System Modelling Canvas

that led to the high impact (and high risk) consequences. It isn't always possible to address the root cause, e.g. because the assets involved may be in a customer network, although in that case one could provide advice to the operator of the affected devices. But if the root cause can be addressed, that is usually the best way to secure the overall system with the least amount of potentially costly security measures. Applying this filter to the example in Figure 4.7 reduces five very-high risk threats to one root cause, a credential stuffing attack to discover user accounts with weak passwords, which is one of the situations described in the OWASP Top 10 threat A2.

Once a threat has been identified, it can be selected and viewed in the threat editor, as shown in Figure 4.8. This gives details of a threat, including the involved assets, the causes and effects of the threat including the direct and secondary effects, and possible control strategies. Users can select controls to complete a control strategy, whose effects are included when the risk level calculation is next updated. In the example shown, one of the solutions for the credential stuffing attack is to impose password quality checks, following the recommendations specified in NIST-800-63n Annex A [14]. This reduces the likelihood of this attack being successful and significantly lowers the associated risk level, so that when the risk levels are recalculated, attention switches from this one very high risk threat to a handful of high risk threats.

The other way to find the threats that most need addressing is to start by checking the misbehaviours of each asset to find those with a high level of risk. Usually there are far fewer high-risk misbehaviours than high-risk threats, although in the current SSM version they can only be inspected for the selected asset (a system wide list is being added for D7.3, triggered by OCC's most recent validation feedback). Clicking on a misbehaviour opens the Misbehaviour Explorer, as shown in Figure 4.9. The Misbehaviour Explorer lists all threats that are direct or indirect causes of a misbehaviour, and the same threat filters are available as in the system wide threat list, so one can quickly identify the root causes. When looking at causes of a specific misbehaviour, it is often helpful to rank based on the likelihood rather than the risk level, since the risk level of each individual threat may be based on other misbehaviours caused. Ranking by likelihood reflects the

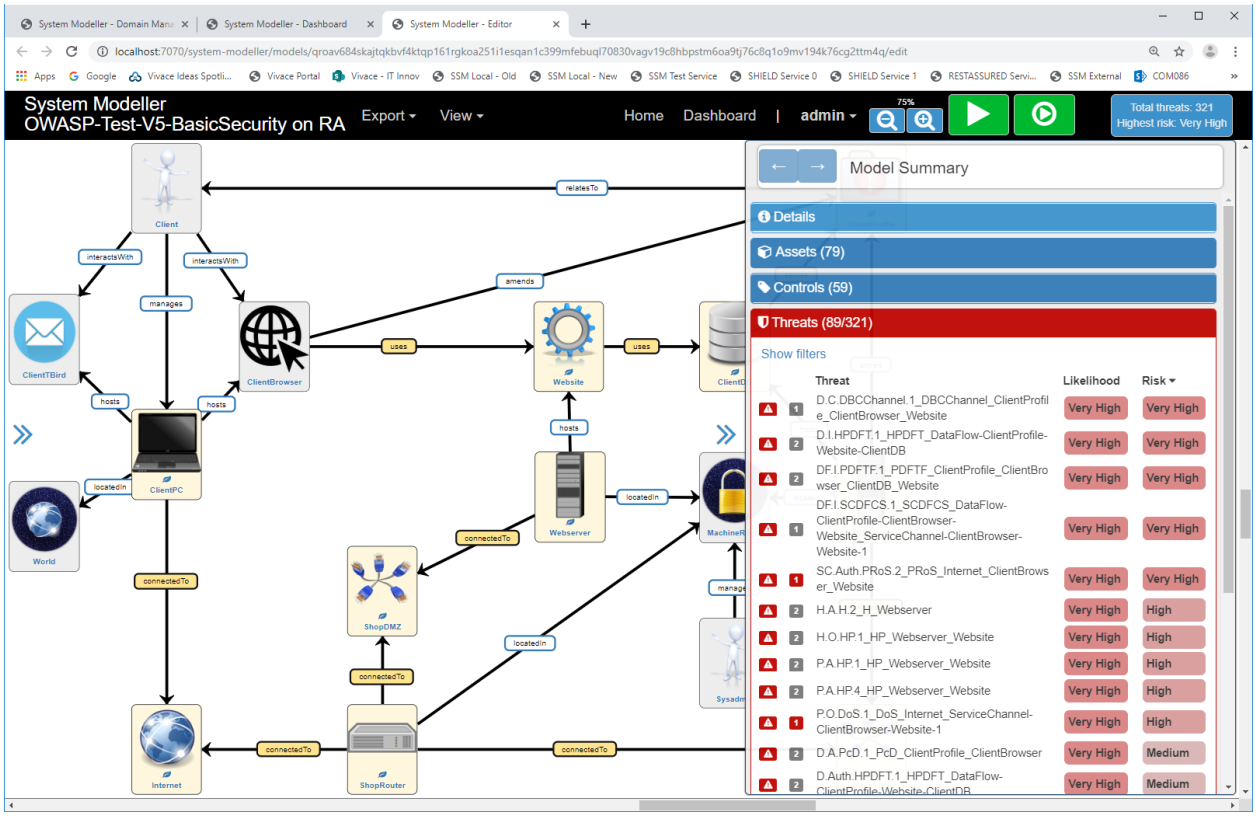

Figure 4.7: The SSM Threat Catalogue

importance of threats to a single misbehaviour.

The Threat Explorer and Misbehaviour Explorer are themselves linked. Clicking on a threat in the Misbehaviour Explorer brings that threat up in the Threat Explorer. Similarly, clicking on a misbehaviour or trustworthiness attribute in the Threat Explorer (a cause or effect of a threat) brings that misbehaviour (or loss of trustworthiness misbehaviour) up in the Misbehaviour Explorer. Thus one can explore attack paths and secondary effect cascades if necessary just by using the two views in combination with each other.

4.2.3 RestAssured enhancements since D7.1

4.2.3.1 Iterative construction rules

As with the domain model described in Section 3.3, improvements have been made in the SSM software in both the SHIELD and RestAssured projects since the release of D7.1. Here we focus on the improvements made in RestAssured, most of which were responses to validation feedback from OCC and Adaptant.

The main algorithmic improvement was the introduction of iterative construction. The changes to the SSM code for this are relatively small, as one just needs to repeatedly apply a construction rule until no further changes occur. Most of the work needed to exploit this was for the creation of rules in the domain model that iteratively construct the desired system model elements (e.g. data paths and data flows), in such a way that the process does eventually terminate. Lessons learned while doing this led to the inclusion of a maximum iteration count for each pattern.

4.2.3.2 Attack path tracing

The other functional improvement was addition of attack path tracing. This is partly algorithmic, but also partly a user interface enhancement.

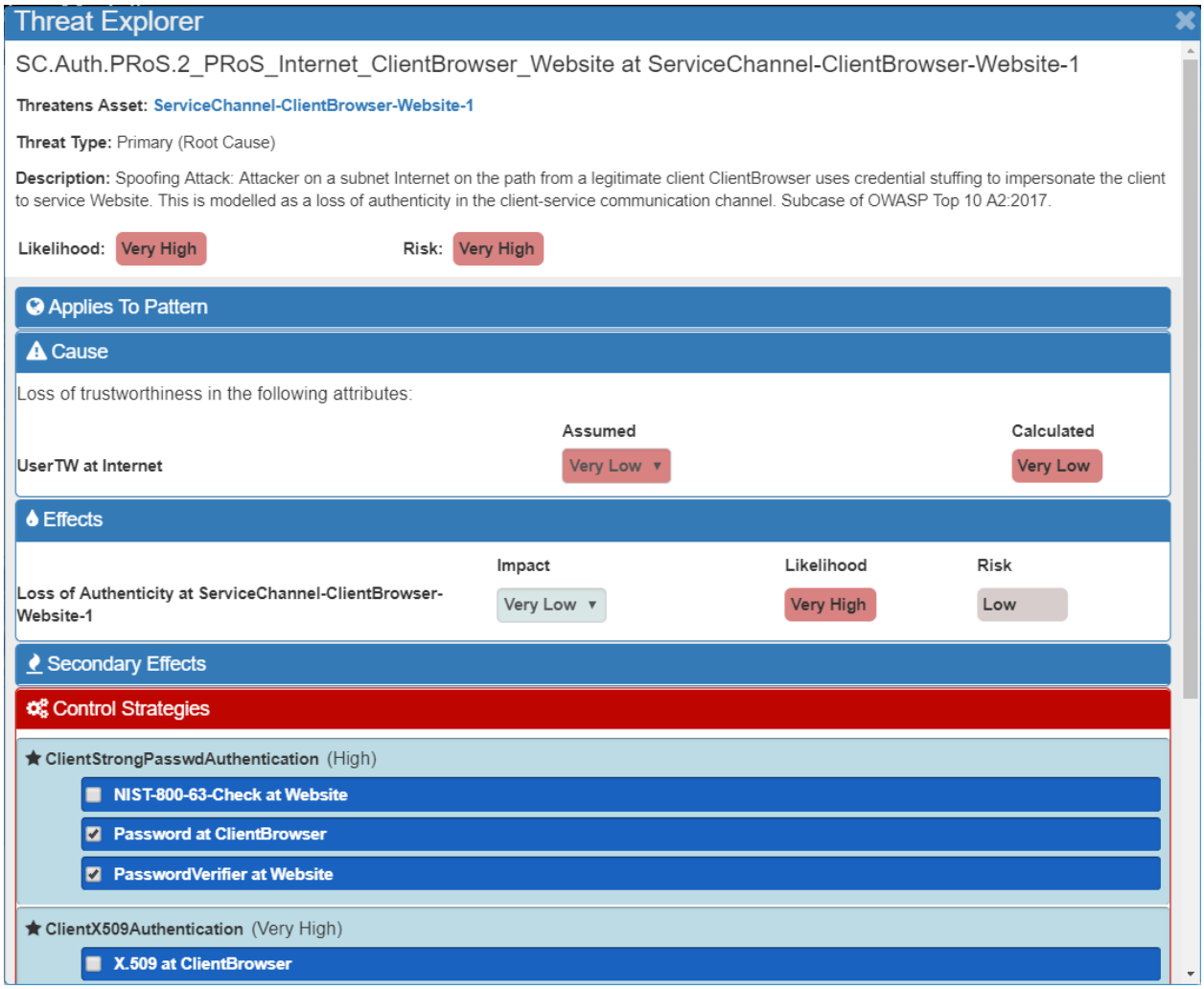

Figure 4.8: The SSM Threat Explorer

In D7.1, SSM supported only secondary effect cascade tracing. The computed risk levels per primary threat were based only on their direct and indirect effects within the secondary effect cascade, ignoring the possibility that these may lower trustworthiness and enable an attack path by increasing the chances of further primary threats. It was also impossible to use the Threat and Misbehaviour Explorers to trace back beyond the last primary threat in each attack path, or to find the true root causes of misbehaviours.

To fix this, links between misbehaviour and trustworthiness were added to the secondary effect cascade tracing code, allowing tracing beyond the last primary threat leading to each misbehaviour to find complete attack paths. This automatically added the full attack paths to the threat listings, whose titles were modified to explain what the listings now contain. For example, in the Misbehaviour Explorer the threat listings are now headed as ‘All Causes’ followed by ‘Root Causes’.

At the same time, an extra threat status flag was introduced for root cause threats (those at the starts of attack paths), and filters added to make use of this information. The Threat Explorer now indicates whether each threat is a root cause, a primary threat somewhere along an attack path, or a secondary threat. These types are also displayed in the threat lists in the main threat catalogue and in the Misbehaviour Explorer, using a ‘2’ symbol for secondary threats, a ‘1’ symbol for primary threats, and colouring this red if the threat is also a root cause that starts an attack path. These new GUI features can be seen in the threat listings in both Figure 4.7 and Figure 4.9.

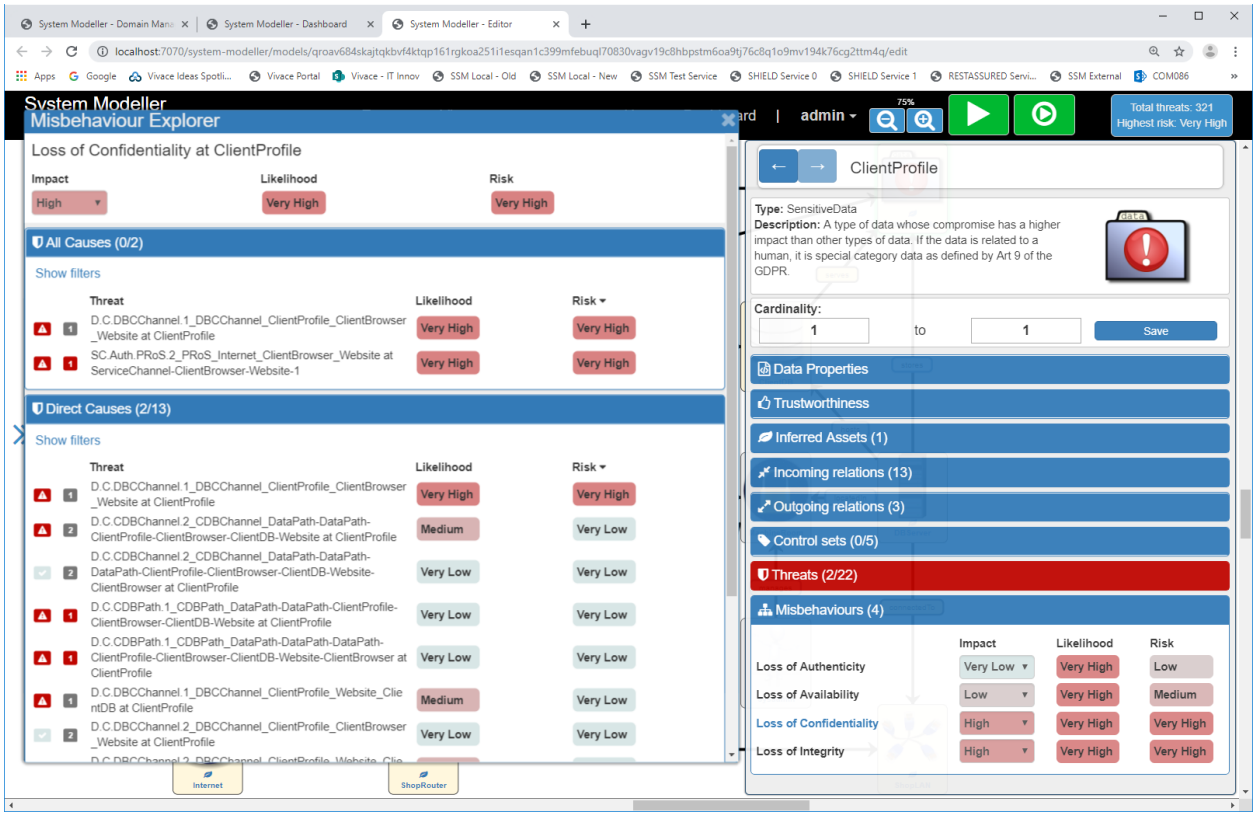

Figure 4.9: Finding Threats using the SSM Misbehaviour Explorer

4.2.3.3 Model report feature

The model report feature shown in Figure 4.10 was also added for RestAssured since D7.1. This was considered the most important of all the feature requests generated by the OCC validation exercise in late 2018 and early 2019. The new feature is accessed via a link in the Model Summary in the right hand panel, which pops up a dialog window summarizing the system model information; this includes assets (along with their trustworthiness values, control sets and misbehaviours), threats (including status, likelihood and risk), and compliance issues. The report can be exported to HTML for easy incorporation into Word documents.

It is not yet clear whether the report should provide a specification for the programmers implementing the system, a risks-versus-cost analysis for senior managers or investors, or a report for compliance auditors tracing the means by which each risk has been addressed. Each of these is useful only to its intended audience. The original plan reflected in the RestAssured DoA was to target senior managers with information about the cost of controls and their effect on risk levels. However, the initial feedback from Adaptant and OCC focused more on information for implementers (who need to implement security measures) or penetration testers (who will verify whether they are effective). This will be resolved with them and the most appropriate report format incorporated into D7.3. Until then, the report is just a simple listing extracted from the model.

4.2.3.4 Performance improvements

The validation by OCC also produced a lot of feedback on usability issues, many of which were really about the performance of SSM. The amount of data transferred was sometimes quite high, and not helped by the inefficient serialisation of RDF graphs in XML (using the NQ format). This did not have much impact when tested at IT Innovation over a high bandwidth local connection, but for remote users at OCC and Adaptant it was a serious concern.

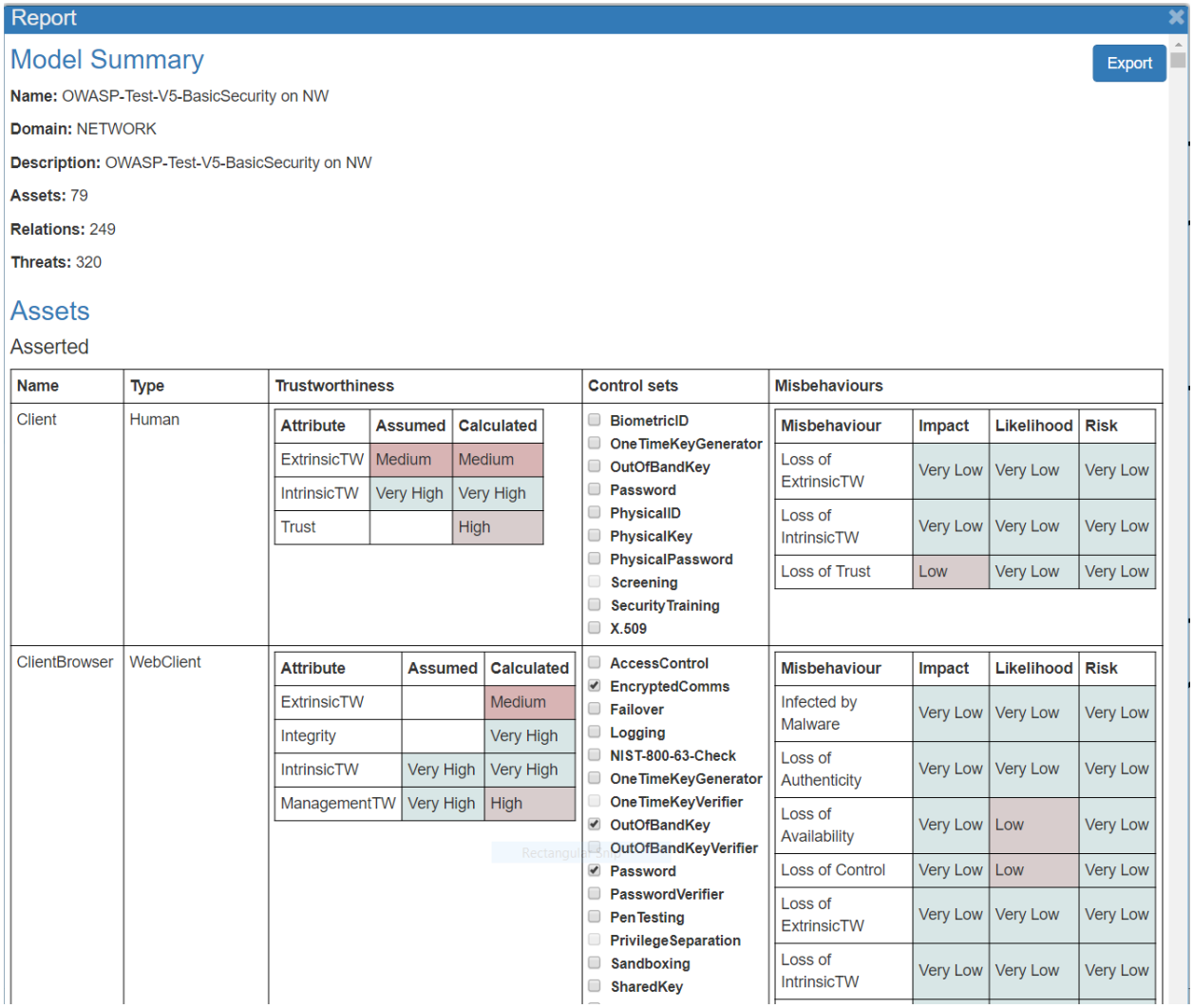

Figure 4.10: The SSM Report Generator

Several actions that should have been simple but were taking too long were optimised, including:

- Moving asset(s) on the canvas;
- Renaming an asset;
- Changing the cardinality of an asset; and
- Changing the type of an asset.

These all took several seconds over a less than perfect connection and were reduced to well under a second per change. Actions requiring more information to be exchanged could not be improved in this way, so data compression was introduced for large data transfers. Those actions (such as loading a model or loading updated threat and risk information after validation and risk evaluation steps) now take a few seconds once the calculation is complete, where previously they could take up to a minute to load new data.

Calculation times for model validation and risk evaluation were also identified as targets for further optimisation. The biggest single contributing factor was the time for validation, which took several minutes for simple models used in the SSM training tutorial, and up to 20 minutes for the largest models produced by WP8 validation partners. Since D7.2, the SSM validator has been reformulated to avoid using any general

purpose semantic reasoning features of the semantic store. Instead, the model is loaded from the triple store into Java data structures, and the algorithms for applying inference rules defined in the knowledge base implemented in Java, reading and writing to and from these Java data structures. Only once the validation process is complete is the model written to the triple store. These changes produced a 100-500 fold improvement in validation times, so that now the only discernible delays are when writing the model back to the triple store, and (if using a slow network connection) when returning results to the GUI.

The next priority for performance optimisation is the risk level calculation, which can take 10-20 seconds for a large model including load and store operations and the GUI refresh. It is expected that some improvement will be possible, and in the meantime the RRE uses a simplified API so far less information has to be exchanged between the SSM and RRE than between the SSM and the SSM GUI.

4.2.3.5 Usability improvements

The SSM model editor main menu has been simplified. Previously it provided access to 'configuration' options most of which involved setting model metadata. This has been moved to the dashboard view where models are listed. At the same time, the model export features from the dashboard view have been replicated in the model editor main menu, as users complained that they wanted to save versions during the model editing process.

Right click options have been added to access context menus from assets and relationship labels. This allows more information to be accessed or in some cases edited directly within the modelling canvas, where previously one had to select an asset and open the relevant section of the properties panel on the right.

Messages are now displayed in the Model Summary in the properties panel to warn the user when there is out of date information and the model needs to be revalidated.

A GUI option has been added to reveal or hide inferred relationships between asserted assets. Some construction rules in the domain model are now intended to fill in relationships that were previously added explicitly by the users. But one consequence of this is that users became unsure which relationships might still be missing. Figure 4.11 shows the same model as 4.6 but with inferred relationships revealed.

Most of these had to be inserted manually in the D7.1 release, but the addition of more construction rules in the D7.2 release means users can now insert a small number of links and the rest will be automatically inferred.

If the canvas window is zoomed out, then when the cursor is placed over an asset icon, the icon will enlarge to its original size making it far more readable. There are also options to access asset or relationship information by right clicking on the selected asset, and to shift the view to either the centre of the canvas or the centre of the model.

When creating links between assets, the user must click a green plus sign on the first asset, then click on the second. If there is only one possible link between the two assets, it is automatically created. If there are several options, a drop-down selector then appears, with tool tips explaining what each type of relationship means. This lists all possible links from the first asset to the second, but now it also lists possible links from the second asset to the first. This was a response to feedback from validation partners who pointed out that they couldn't always remember the direction a link should have.

A feature for 'hiding' asset relationships in the canvas has been added, allowing relationships that cross the entire canvas to be removed from view, making it easier to see relationships that represent local interdependencies. This is typically used when a client on one side of the canvas uses data stored in a host or database on the other side of the canvas. The connection between the two may cross multiple other relationships and make them harder to follow by eye, so temporarily removing it from the display can be very helpful. Functions have also been added to show previously hidden relationships, and also to show any inferred relationships, helping users to understand what can be inferred and what they may need to add to a system model.

One problem raised by OCC is that it can be very difficult to understand which controls would help to

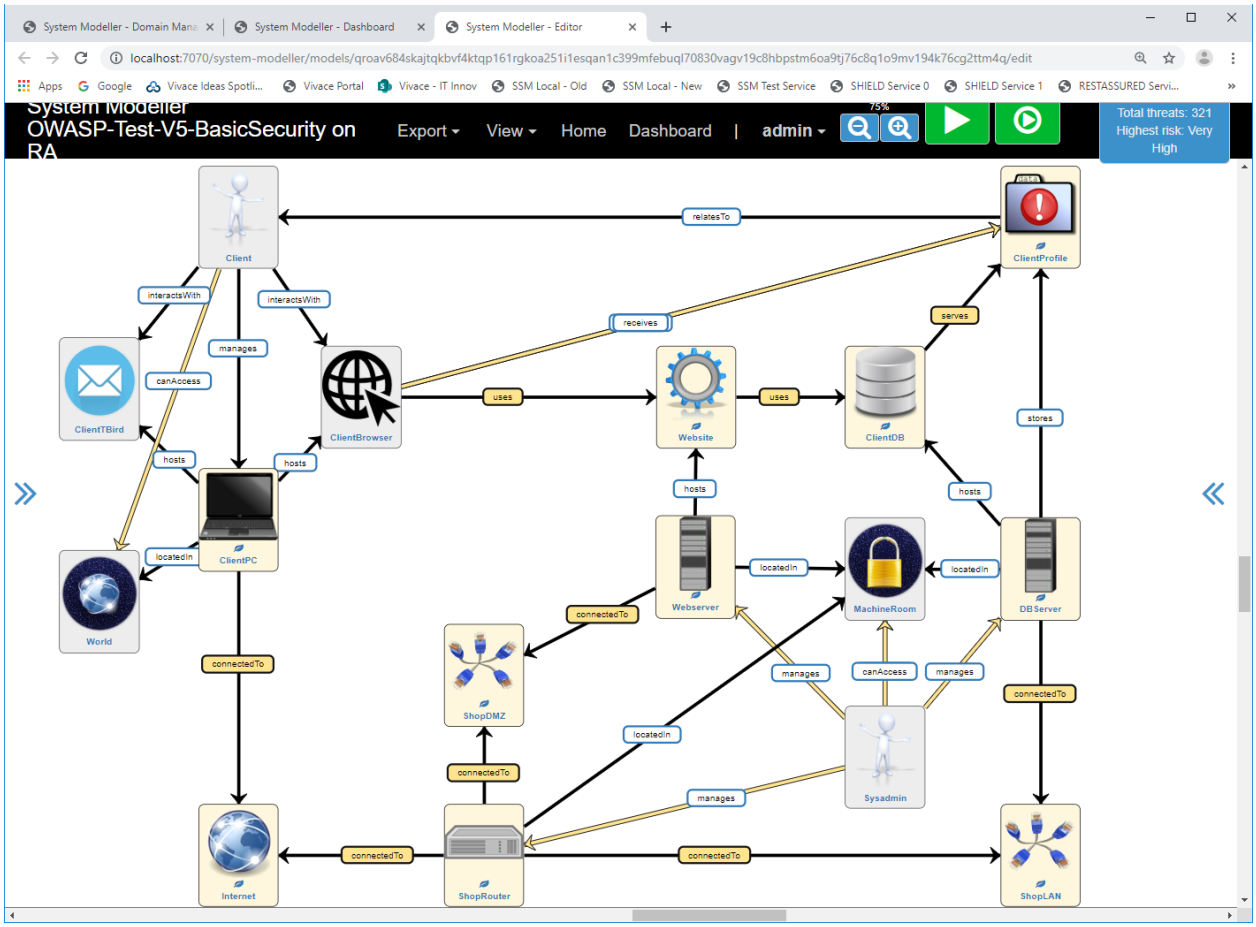

Figure 4.11: Displaying Inferred Relationships

reduce risk levels when the number of threats contributing to those risks is very large. The threat list interface does provide filtering functions (e.g. to show only threats with control strategies or the first threat in each attack path). But early in the iterative ISO 27005 risk assessment procedure there will be few controls and hundreds of threats may be rated as high risk threats. To address this a new feature was introduced since D7.2 providing a system level view of risks per misbehaviour (i.e. potential threat consequence) rather than per threat. Because the number of consequences is always far smaller than the number of threats that could produce those consequences, this provides a far shorter list of starting points from which to look for control strategies that should be used.

Finally, tooltips and other such messages have been improved. For example, the tooltip for the model validation button now says 'Validate Model and Find Threats', which is more meaningful to users than simply 'Validate Model'. Some tooltips have also been added, e.g. where fractions appear in threat listings, a tooltip is now attached to explain that '1(1/8)' means '1 out of 8 addressed'.

4.2.3.6 Bug fixes

Many bugs found in D7.1 were fixed in D7.2, including:

- Add button in Outgoing Relations brought up Add Incoming Relation dialog by mistake.
- User interface overlay used when adding a relationship disappeared too early when multiple relationships are added.
- There was no limit on asset name length.

- Models became corrupted if an asset is moved when it is being deleted.
- Default impact levels were not set correctly in some circumstances.
- User asserted impact levels were incorrectly overwritten in some circumstances.

N.B. Certain reported bugs relate to the use of the tool in the Safari browser. These bugs have not been addressed and Chrome remains the target browser for now, as it is available on all platforms.

The build process used for SSM components was also improved, supporting continuous integration and the inclusion of more tests and static code analysis using SonarQube.

For D7.3, a simplified knowledge base was developed to tests the reformulated SSM validator, and this allowed further bugs to be identified affecting the previous validator which have now been fixed in the optimised version. In the last few months of the project, the SSM software has become significantly faster, easier to use, and also more accurate in its application of machine reasoning to implement the models described in Section 3.2.

4.3 Run Time Risk Evaluator

4.3.1 RRE design

The Run-Time Risk Evaluator (RRE) tool has now been added to the risk analysis tool set from WP7. This did not exist in D7.1, although a very early prototype was demonstrated at the 1st Review in July 2018. At that time, it demonstrated the concept of calculating risk levels, but using risk levels pre-calculated by SSM.

Since then, the specification of the RRE tool has evolved. Previously it was assumed there would be two separate tools for run-time risk monitoring and run-time risk assessment. Now it is clear that the adaptation system from WP5 will interface with monitoring systems, and provide all information on both detected and proposed system changes. This change removes the need for two separate services to handle the change monitoring and change proposal channels.

The procedure for using the RRE has also been refined, addressing three main issues:

- the desire to combine the run-time risk monitor and run-time risk evaluator services into one service;
- the interpretation of design time SSM system models, given the changes in the workflow between SSM and ReAs-CSAP; and
- the need to manage performance of risk calculations, which at the time of the 1st review would have forced the use of pre-calculated risk levels had the necessary algorithm been integrated.

System models created by SSM are design-time models. By this we mean that even if created after a system has been deployed, and SSM model is not supposed to capture the current configuration, but the range of possible configurations, along with all the attendant future risks. For example, if a system includes both primary and back up query servers, both should be included in the SSM system model so any deficiency in the security of either server will show up as a potential source of risk. At run-time, the system will be using only one of these servers, usually the primary server but occasionally the back up. The run time risk level calculation therefore doesn't need to consider both servers and should always represent a subset of the design time model.

The performance of risk level calculations in SSM has improved significantly since D7.1. Some of this improvement comes from using a run-time API that transfers less data than the API used to support the SSM GUI. Calculation speed has also improved as lessons have been learned on how to refine the domain model, which now uses more up-front construction rules but simpler threat patterns. The new validator added for the D7.3 release produced over 100-fold improvement in validation performance. Risk level calculations also

benefited from these improvements, although not to the same extent as the risk level calculation algorithm isn't was not upgraded. It is expected that risk level calculations will continue to get faster, although model validation is probably now as fast as it can be.

It was decided that the RRE should use SSM as a risk level calculator service, with pre-validated models representing different potential variations in the system topology. Validation with a new system topology will probably never take less than a few seconds, but risk level calculation is likely to get faster, and since the RRE does not need to access the entire threat catalogue it is fast enough for a demonstrator.

This leads to the following procedure:

1. Create the design-time SSM model capturing all the asset types that may become part of the system, validate and evaluate risks, and use this to inform the specification of security requirements.
2. Identify variation points embedded in the design-time model, that is to say:
    - parts of the model that may be inactive, such as back up servers;
    - parts of the model that may be merged, e.g. if services that may be running in different jurisdictions are at some point in the same jurisdiction.
3. Create and validate SSM models corresponding to each possible variation that may be used by system adaptation.

Previously it was felt that in Step 2 above, variants may be needed for different combinations of security controls, or different trustworthiness assumptions (e.g. moving from a trusted hybrid cloud service provider to a less trusted public cloud). However, it now seems that it should be possible to update the risk levels fast enough when these inputs change, so pre-validated variants will only be needed for different system topologies.

The RRE can then be started, giving it references to the set of pre-validated variants in SSM, each of which will have a threat catalogue and calculated risk levels with the security measures defined in the design-time model. At this point, system context from ReAs-CSAP can be used to decide which variation represents the initial deployment configuration.

From this point onwards, the adaptation component from WP5 should request a risk level update when either the system configuration has changed, or a change in the system configuration (i.e. an adaptation)is proposed. The RRE service must examine the change and decide which topology variant provides the best match. Then it must pass any changes in security control status or trustworthiness assumptions to SSM, applying them to each variant where the associated asset type is present. That step may be handled via the Risk Dashboard, which could also be used to enter changes that cannot be detected by system monitoring and must be supplied by a user. Changes must be applied to every variant in which they are relevant because the system may change topology in future, and all variants must be kept up to date. Then the RRE can request a risk level calculation using the variant that best represents the current or proposed system configuration and return the results to the adaptation component.

The final components are still under development. The overall configuration is expected to be as shown in Figure 4.12. The base model is the design time SSM model incorporating all expected variants. Other models are subsets representing one possible variant, one of which will be the variant that best represents the current configuration. Other variants represent possible future configurations that may be reached after adaptations.

4.3.2 Workflow between the adaptation service and RRE

APIs have now been defined together with WP5, and developed for the RestAssured services that are concerned with:

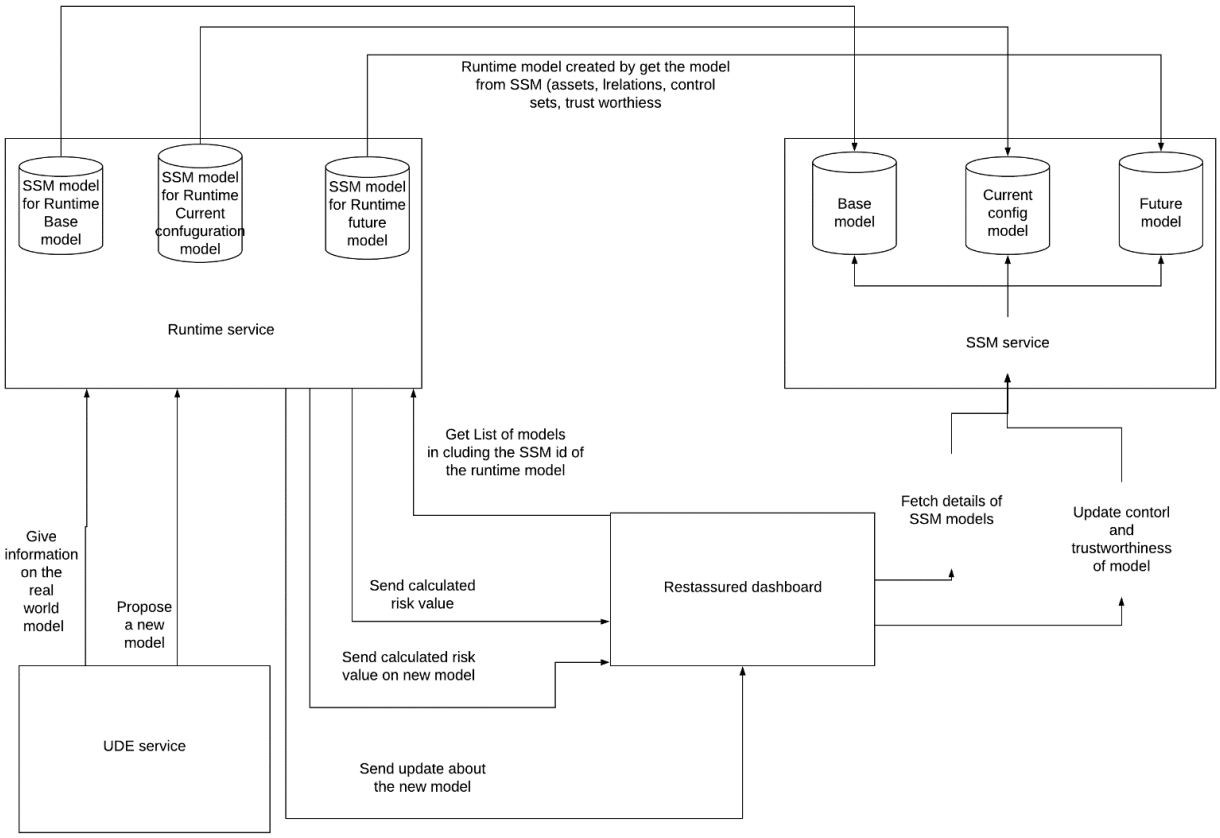

Figure 4.12: WP5-WP7 Interfaces

- maintaining and adapting the current runtime system (the adaptation from WP5, developed by UDE); and
- evaluating runtime system risk levels (the runtime risk evaluator from WP7, developed by IT Innovation).

Overall, the UDE Adaptation Service sends a runtime model (in a format defined by UDE) to the RRE, which must then analyze this model in order to determine its threat catalogue and calculated risk levels. According to the evaluated risks, the RRE then notifies the Adaptation Service, to say either that the runtime model's risks are below an acceptable threshold, or to indicate that the risk level is too high, and provide information on the threats identified as being responsible.

Note that compliance threats (see Section 3.2.6.7) have no risk level, because they represent a potential breach of regulations which is either present or not present. If the SSM risk level calculator finds that there are unaddressed compliance threats in the new or proposed configuration, it acts as though the risk level is too high, sending information about the identified compliance threat.

This workflow is shown in Figure 4.13.

The service workflow starts with an initial registration process, in which the RRE notifies the Adaptation Service of its service endpoint, to which subsequent requests for risk evaluation, etc., can be sent. After some form of monitoring event, the Adaptation Service sends the updated runtime model to the RRE for evaluation ("updateRuntime"). The body of this request contains the runtime model itself, the schema for which was defined by UDE in WP5, using JSON format for serialisation and inclusion in the request message.

The RRE analyses the runtime model for risks (as described above). If the risk levels are considered

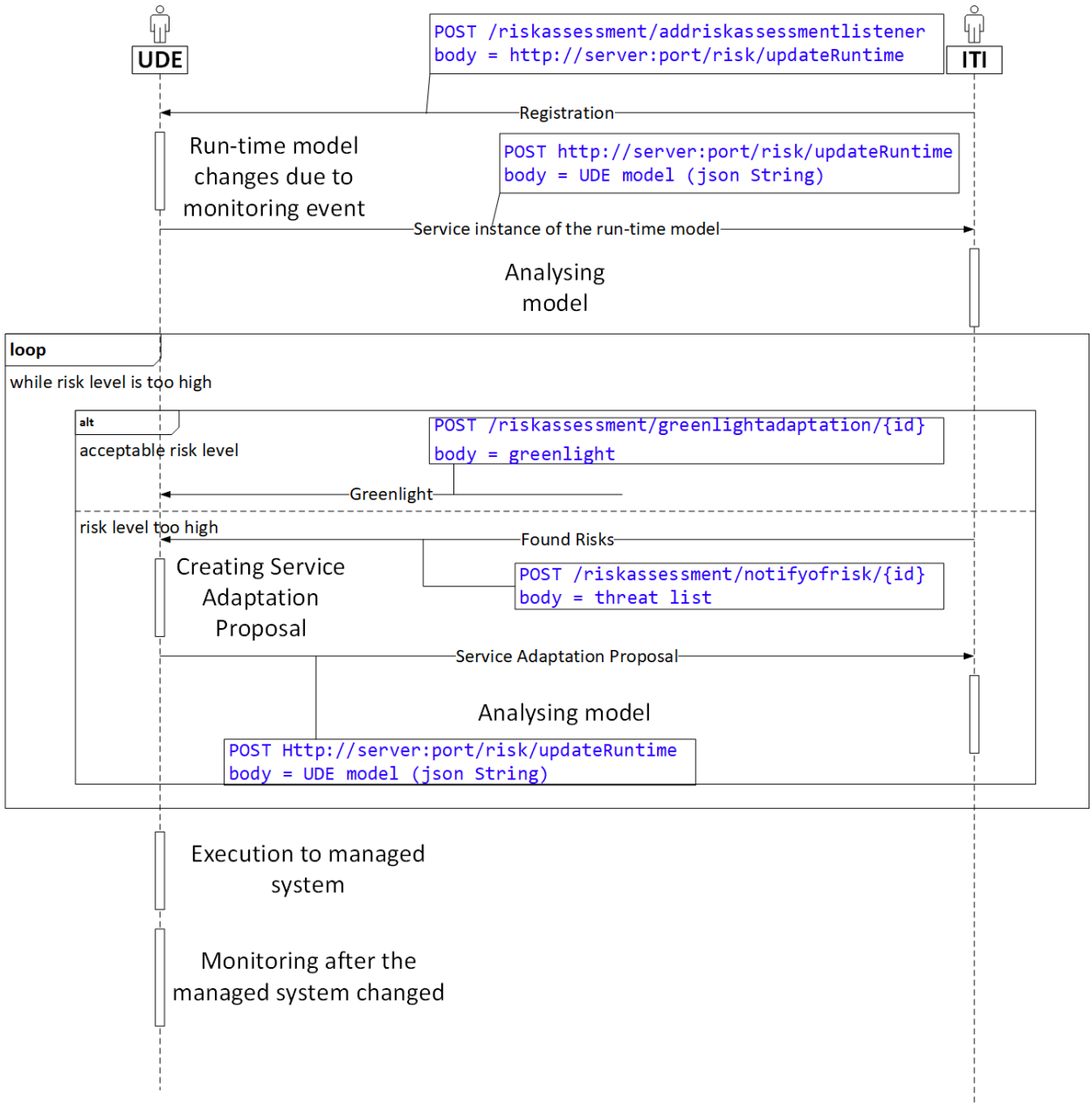

Figure 4.13: WP5-WP7 Workflow

acceptable, a ”greenlightadaptation” request is sent back to the Adaptation Service, which indicates that this runtime model is OK, and no changes are required.

If the risk levels are found to be too high, the RRE sends a ”notifyofrisk” request to the Adaptation Service. The body of this request contains a list of active threats (or compliance threats), along with hints (derived from SSM control strategies for these threats) on how they may be addressed. The Adaptation Service will then try to find a system adaptation to address the threats and reduce the risks.

Whenever the Adaptation Service wishes to implement an adaptation (for any reason, not only because the RRE says the risk level is too high), the same procedure is used predict risk levels after the proposed change. In that situation, the runtime model supplied to the RRE includes the proposed change. The Adaptation Service may make any number of adaptation proposals, until it finds one that gets a green light from the RRE.

4.3.3 Mapping between the runtime model and SSM model variants

In order for the RRE to analyze a runtime model sent from the Adaptation Service, it is first necessary to map from the UDE runtime format (JSON) into a common Java object structure that then enables direct comparison with system models stored in the SSM. Using the common structure allows the RRE to match an incoming UDE runtime model to one of the possible SSM system model variants, whose risks can then be evaluated and returned to the Adaptation Service (or a ”greenlight”, as appropriate).

The initial UDE runtime model is sent (via the ”updateRuntime” request) in a JSON format such as the following:

```
{
  "type":"CloudEnvironment",
  "@id":"/",
  "tosca_nodes_root":[
    {
      "type":"Compute",
      "@id":"1",
      "name":"DBServer_1",
      "id":1,
      "hosts":[
        {
          "type":"DBMS",
          "referencedObjectID":"8"
        },
        {
          "type":"SoftwareComponent",
          "referencedObjectID":"35"
        }
      ],
      "connectedToNetwork":[
        {
          "type":"Network_Network",
          "referencedObjectID":"2"
        }
      ],
      "jurisdiction":"DE"
    },
    {}...
  ]
}
```

Here, the important data is in the ”tosca nodes root” entry, which contains an array of nodes (known as ”assets” in the SSM). Each node has a ”type”, ”@id” and ”name” (N.B. ”id” fields are ignored). These common fields clearly define the type of the node/asset, its unique id and name. Other fields may be defined, such as ”jurisdiction”, which apply to certain node/asset types only. The remaining fields define the connections to other nodes/assets in the model. In this example, we have ”hosts” connections to two other nodes, with ids ”8” and ”35”. The types of these nodes are also defined, which may also be used in subsequent mapping, as only certain node types may be connected, with specific connection types (also known as ”relations” in the SSM model).

The UDE model is loaded and parsed into an ”SSMModel” object in the RRE; this contains equivalent fields for name, type, id, etc. and has a list of assets and their connections to other assets (i.e. “relations”).

The SSM model variants will already have been pre-loaded into the RRE, either programmatically or via a REST call to one of its API endpoints. These are also stored in memory as SSMModel objects, so are

available for comparison with incoming UDE models as soon as they come in.

When a new UDE runtime model arrives in the RRE, it is parsed (as described above) then compared in turn to each of the SSM model variants. Once a match has been found, the ”updateRuntime” method returns with a simple 200 HTTP response, then the RRE determines the risks for this model, as described earlier. The following section describes the mapping process in more detail.

Firstly, we have agreed and defined a static mapping of UDE asset types to those used in the SSM. In some cases, these asset types match (e.g. ”DBMS” is found in both), however most require a mapping, e.g. ”Database” maps to ”DB” in the SSM. Some cases have multiple mapping options, e.g. ”Compute” could map to either ”VM” or ”Workstation” in the SSM.

We have also agreed that, where possible, assets in the UDE model will adopt the same name as its equivalent in the SSM model. This helps to remove ambiguities where there may be several assets of the same type in the system, and of course speeds up the matching.

One issue that complicates matters is that the UDE runtime model may contain multiple instances of a particular asset, which would map to a single asset in the SSM model. For example, an incoming UDE model may have assets called ”DBServer 1” and ”DBServer 2”. Both are active database servers, however only one of these is currently of relevance in the system, as defined by its connections to other assets. This means that we generally need to map one of these instances to its equivalent in the SSM, which will be simply named ”DBServer” (we have also agreed that, where multiple instances exist in the runtime model, these will be named with a numerical suffix.

A further complication is that certain UDE assets do not have a formal equivalent in the SSM model (or vice versa). For example, the SSM has a ”Jurisdiction” asset type, whereby one or more other assets can be connected to this in order to indicate that they are located within this jurisdiction. However, the UDE runtime model simply has a ”jurisdiction” field for certain asset types. In this case, when parsing the UDE runtime model, the RRE creates inferred assets (in the received runtime model) to represent jurisdiction assets that should then match to an equivalent in the SSM. In this way, the RRE can compare assets and relations between the models correctly.

The complete mapping process is somewhat iterative and complex, as some assets can be mapped easily at the start, but others may only be mapped once assets they have connections with have been mapped, allowing the RRE mapper to determine which of several instances is currently active (as described above). Once the active instances have all been identified, the remaining instances are discarded from further mapping, as they are considered irrelevant.

If after this there are any remaining non-mapped assets, it means the incoming UDE runtime model does not in fact match the SSM variant being compared. This should only happen if the runtime model contains nodes that do not correspond to any asset class defined in the SSM model.

5 Conclusion and Future Work

In this deliverable we have reported the outcomes from WP7 since the start of the second reporting period. The major achievements were:

- Developing an overall methodology for using these tools to manage risks in cloud-based applications that is aligned with the ISO 27001 and ISO 27005 standards for information security risk management procedures.
- Extending the SSM and CSAP approaches to support this methodology and addressing requirements from RestAssured WP8 validation case studies from Adaptant and OCC.
- Refining the theoretical underpinnings for these two design-time tools, and developing procedures for using them to assess risks prior to application deployment, and to prepare models for run-time risk evaluation.
- Developing a new SSM domain model (knowledge base) for RestAssured, addressing issues raised by the validation studies, and reducing the potential for users to overlook important assets and dependencies.
- Enhancing the two design-time tool prototypes, including extensive performance optimisation of the SSM tool in preparation for its use in support of run-time risk evaluation.
- Creating mappings between the models used by CSAP and SSM, and between SSM models and the runtime system model from RestAssured WP5.
- Extending the SSM knowledge base to remove ambiguities in the mapping to the runtime system model, allowing an automated mapping algorithm to be developed and used to map between WP7 risk models and WP5 system models at runtime.
- Developing an initial prototype of the RRE tool, incorporating this mapping algorithm and using SSM automated risk level calculation algorithms at runtime.
- Engaging with WP8 validation partners and supporting their validation scenarios, providing assistance for the novel risk modelling approach in their applications, and responding to feedback to improve the utility and usability of design-time tools.

The philosophy adopted is that security and privacy must be addressed 'by design' (as required by the GDPR Article 25), yet managed and maintained also during run-time. The focus is therefore to capture information relevant to security and data protection risks prior to deployment, using tools to analyse security requirements at that stage, and create machine understandable models allowing automated threat identification and risk level evaluation. The workflow for doing this starts with the developer of application software and services, using the SSM tool to capture the architecture and dependencies of the system. The CSAP tool is then used by the service operator to add the context for a specific deployment. This is exported back to the SSM and RRE tools so it can be used as the starting point for run-time risk evaluation. To achieve this, a mapping has been created between the information used in each tool, which is now being updated as details of the models used are iteratively refined.

The SSM tool uses a knowledge base known as a domain model, describing the types of assets and security related attributes, and potential threats and consequences. This has been extensively revised to address concerns raised by the validation case studies, coming from the validation partners (e.g. the need to explicitly address well known customer concerns such as the OWASP Top 10 threat list), and also from observation of the validation partner experiences (e.g. the need to increase the amount of machine inference

used to simplify the creation of system models by SSM users). Improvements have also been made in the analysis of the resulting models, most notably by the addition of attack path tracing algorithms making it much easier to identify and address root causes of disruptions found in the model.

The SSM tool has also been improved in other ways in response to validation partner feedback, mostly to improve usability and performance. The performance improvements are also motivated by a desire to use the SSM's automated risk level evaluation algorithms at run-time. Improvements made for the D7.3 release have produced better than 100-fold improvement in processing time for model validation (the most computationally intensive step used in the analysis). This was achieved by minimising the use of general purpose semantic reasoning technology in the back end triple store, by moving the data into Java objects and implementing special-purpose reasoners to work with those objects before writing results back to the store. At the time of writing this approach is being extended to other computationally intensive steps including risk level calculation. While the same level of performance improvement is not expected here, it should be enough to support the runtime risk level updates needed by the Runtime Risk Evaluator.

In the final period of the project, one focus has been on the methodology for 'security and privacy by design'. This was revised during the second half of the project to include support for software/service supply chains, and outlined in D7.2. The procedure has now been elaborated in detail, including mapping between the methodology and the ISO 27005 standard, and ensuring it does address the needs of the service operators and their software/service suppliers.

The other main area of work was to integrate the SSM, RRE and WP5 Adaptation Service components needed to implement runtime risk level updates so they can be used by the Adaptation Service to trigger or guide autonomic management of the running cloud based application. From the WP7 side, much of the work was on the mapping procedure to translate between the runtime system model schema used in WP5 and SSM models coming from the pre-deployment 'security and privacy by design' procedure.

Both these areas of work required and benefited from further improvements in the SSM software and knowledge bases. The knowledge base has been augmented to include extra assets to simplify the mapping between an SSM model and the runtime system model schema from WP5, and also to model adaptation control strategies which a designer may choose to depend on to manage some risks, and which provide the basis for sending hints to the WP5 Adaptation Service on what it might do to address high risk threats. The new, fast validator was also integrated for the D7.3 release of SSM, improving computation time to a level that can be used in the loop of the autonomic runtime management system.

The end to end workflow for risk assessment and risk management using the RestAssured methodology has now been demonstrated, starting with the software/service supplier's security and privacy by design procedure supported by SSM, moving to the service operator's use of CSAP to capture context and integrate it with a risk model to complete this process prior to deployment, and ending with the RRE tool providing updates to the risk level estimates at runtime, supporting risk assessment for adaptations used for autonomic system management in a cloud based system or application.

Bibliography

[1] I. Alexander. Misuse cases help to elicit non-functional requirements. *Computing and Control Engineering*, 14:40–45(5), February 2003.

[2] Ittai Anati, Shay Gueron, Simon P. Johnson, and Vincent R. Scarlata. Innovative Technology for CPU Based Attestation and Sealing. In *Proc. of the 2nd Intl. Workshop on Hardware and Architectural Support for Security and Privacy, HASP*, 2013.

[3] Kristian Beckers, Isabelle Côté, and Ludger Goeke. A catalog of security requirements patterns for the domain of cloud computing systems. In *Symposium on Applied Computing, SAC 2014, Gyeongju, Republic of Korea - March 24 - 28, 2014*, pages 337–342, 2014.

[4] Kristian Beckers, Isabelle Côté, Ludger Goeke, Selim Güler, and Maritta Heisel. A structured method for security requirements elicitation concerning the cloud computing domain. volume 5, pages 20–43, Hershey, PA, USA, April 2014. IGI Global.

[5] Kristian Beckers, Holger Schmidt, Jan-Christoph Küster, and Stephan Faßbender. Pattern-based support for context establishment and asset identification of the ISO 27000 in the field of cloud computing. In *Sixth International Conference on Availability, Reliability and Security, ARES 2011, Vienna, Austria, August 22-26, 2011*, pages 327–333, 2011.

[6] Lawrence Chung. Dealing with security requirements during the development of information systems. In *Advanced Information Systems Engineering, CAiSE'93, Paris, France, June 8-11, 1993, Proceedings*, pages 234–251, 1993.

[7] Regulation 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). *Official Journal of the European Union*, L119:1–88, May 2016.

[8] Stefan Fenz, Gernot Goluch, Andreas Ekelhart, Bernhard Riedl, and Edgar Weippl. Information Security Fortification by Ontological Mapping of the ISO/IEC 27001 Standard. In *13th Pacific Rim International Symposium on Dependable Computing (PRDC)*, pages 381–388, 2007.

[9] Eduardo B. Fernandez, Raul Monge, and Keiko Hashizume. Building a security reference architecture for cloud systems. *Requirements Engineering*, 21(2):225–249, Jun 2016.

[10] Eduardo B. Fernandez, Nobukazu Yoshioka, Hironori Washizaki, and Madiha H. Syed. Modeling and security in cloud ecosystems. *Future Internet*, 8(2), 2016.

[11] Donald Firesmith. Specifying reusable security requirements. *Journal of Object Technology*, 3(1):61–75, 2004.

[12] Rune Fredriksen, Monica Kristiansen, Bjørn Axel Gran, Ketil Stølen, Tom Arthur Opperud, and Theo Dimitrakos. The coras framework for a model-based risk management process. In Stuart Anderson, Massimo Felici, and Sandro Bologna, editors, *Computer Safety, Reliability and Security*, pages 94–105, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[13] Ludger Goeke, Nazila Gol Mohammadi, and Maritta Heisel. Context analysis of cloud computing systems using a pattern-based approach. *Future Internet*, 10(8):72, 2018.

[14] P. L. Grassi, J. L. Fenton, E. M. Newton, R. A. Perlner, A. R. Regenscheid, W. E. Burr, J. P. Richer, N. B. Lefkovitz, J. M. Danker, Y. Choong, K. K. Greene, and M. F. Theofanos. Nist special publication 800-63b: Digital identity guidelines, 2017.

[15] Charles B. Haley, Robin C. Laney, Jonathan D. Moffett, and Bashar Nuseibeh. Security requirements engineering: A framework for representation and analysis. *IEEE Trans. Software Eng.*, 34(1):133–153, 2008.

[16] Intel. https://software.intel.com/en-us/sgx/details. Accessed: 2018-07-13.

[17] International Organization for Standardization (ISO). Risk Management - Guidelines (ISO 31000:2018), 2018.

[18] International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC). Information technology - Security techniques - Evaluation criteria for IT security (ISO/IEC 15408-1:2009), 2009.

[19] International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC). Information technology - Security techniques - Code of practice for information security controls (ISO/IEC 27002:2013), 2013.

[20] International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC). Information technology - Security techniques - Information security management systems - Requirements (ISO/IEC 27001:2017), 2013.

[21] International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC). Information technology - Security techniques - Code of practice for protection of personally identifiable information (PII) in public clouds acting as PII processors (ISO/IEC 27018:2019), 2014.

[22] International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC). Information technology - Security techniques - Information security risk management (ISO/IEC 27005:2018), 2018.

[23] Umar Mukhtar Ismail, Shareeful Islam, Moussa Ouedraogo, and Edgar Weippl. A framework for security transparency in cloud computing. *Future Internet*, Feb 2016.

[24] Michael Jackson. *Problem Frames: Analyzing and Structuring Software Development Problems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[25] M. Kis. Information security antipatterns in software requirements engineering. In *Proceedings of the 9th Conference on Pattern Language of Programs*, 2002.

[26] K. G. Kogos, K. S. Filippova, and A. V. Epishkina. Fully homomorphic encryption schemes: The state of the art. In *2017 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, pages 463–466, Feb 2017.

[27] Sascha Konrad, Betty H.C. Cheng, Laura A. Campbell, and Ronald Wassermann. Using security patterns to model and analyze security requirements, re'03 international workshop on requirements for high assurance systems, usa, 2003.

[28] Tong Li, Jennifer Horkoff, and John Mylopoulos. Integrating security patterns with security requirements analysis using contextual goal models. In Ulrich Frank, Pericles Loucopoulos, Óscar Pastor, and Ilias Petrounias, editors, *The Practice of Enterprise Modeling*, pages 208–223, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[29] L. Lin, B. Nuseibeh, D. Ince, and M. Jackson. Using abuse frames to bound the scope of security problems. In *Proceedings of the 12th IEEE International Requirements Engineering Conference (RE)*, 2004.

[30] L. Lin, B. Nuseibeh, D. C. Ince, M. Jackson, and J. D. Moffett. Analysing security threats and vulnerabilities using abuse frames, 2003.

[31] Alexander V. Lyubimov, Dmitry V. Cheremushkin, Natalia Andreeva, and Sergey Shustikov. Information security integral engineering technique and its application in ISMS design. In *Sixth International Conference on Availability, Reliability and Security, ARES 2011, Vienna, Austria, August 22-26, 2011*, pages 585–590, 2011.

[32] John McDermott and Chris Fox. Using abuse case models for security requirements analysis. In *Proceedings of the 15th Annual Computer Security Applications Conference*, ACSAC '99, pages 55–, Washington, DC, USA, 1999. IEEE Computer Society.

[33] Andreas Metzger, Zoltan Mann, Stefan Schoenen, Paul Mundt, Adam Burns, Mike Surridge, Ludger Goeke, Maritta Heisel, Nazila Gol Mohammadi, Pete Maynard, and Eliot Salant. Restassured deliverable d3.1: Initial high level architecture. Technical report, 2017.

[34] Raydel Montesino and Stefan Fenz. Information security automation: How far can we go? In *Sixth International Conference on Availability, Reliability and Security, ARES 2011, Vienna, Austria, August 22-26, 2011*, pages 280–285, 2011.

[35] National Institute of Standards and Technology(NIST). Security and Privacy Controls for Federal Information Systems and Organizations , 2013.

[36] Yannick Naudet, Nicolas Mayer, and Christophe Feltus. Towards a systemic approach for information security risk management. In *11th International Conference on Availability, Reliability and Security, ARES 2016, Salzburg, Austria, August 31 - September 2, 2016*, pages 177–186, 2016.

[37] Owasp top 10 - 2017: The ten most critical web application security risks, 2017.

[38] S. C. Phillips, M. Surridge, and J. Papay. European security in health data exchange (shield), deliverable 4.4: Privacy by design models and tools: final release plus documentation., 2019.

[39] Holger Schmidt. Threat- and risk-analysis during early security requirements engineering. In *ARES 2010, Fifth International Conference on Availability, Reliability and Security, 15-18 February 2010, Krakow, Poland*, pages 188–195, 2010.

[40] Markus Schumacher. *Security Engineering with Patterns: Origins, Theoretical Models, and New Applications*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003.

[41] Guttorm Sindre, Donald G. Firesmith, and Andreas L. Opdahl. A reuse-based approach to determining security requirements. In *In Proc. 9th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'03*, pages 16–17, 2003.

[42] Mike Surridge, Bassem I. Nasser, Xiaoyu Chen, Ajay Chakravarthy, and Panos Melas. Run-time risk management in adaptive ICT systems. In *2013 International Conference on Availability, Reliability and Security, ARES 2013, Regensburg, Germany, September 2-6, 2013*, pages 102–110, 2013.

[43] Mike Surridge, Toby Wilkinson, Stefanie Wiegand, Ludger Goeke, and Nazila Gol Mohammadi. Restassured deliverable d7.1: Restassured security and privacy engineering methodology. Technical report, 2018.

[44] Yinghui Zhang, Xiaofeng Chen, Jin Li, Duncan S. Wong, Hui Li, and Ilsun You. Ensuring attribute privacy protection and fast decryption for outsourced data security in mobile cloud computing. *Inf. Sci.*, 379:42–61, 2017.