Deliverable D3.2
First High-Level Architecture & Methodology
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.
## RestAssured Consortium


<table>
<thead>
<tr>
<th>Project Number</th>
<th>731678</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Title</td>
<td>RestAssured - Secure Data Processing in the Cloud</td>
</tr>
<tr>
<td>Deliverable Type</td>
<td>External Report</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Title of Deliverable</th>
<th>First High-Level Architecture &amp; Methodology</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nature of Deliverable</td>
<td>External Report</td>
</tr>
<tr>
<td>Dissemination Level</td>
<td>Public</td>
</tr>
<tr>
<td>Contractual Delivery Date</td>
<td>30 April 2018</td>
</tr>
<tr>
<td>Actual Delivery Date</td>
<td>30 April 2018</td>
</tr>
<tr>
<td>Contributing WPs</td>
<td>WP3</td>
</tr>
<tr>
<td>Editor(s)</td>
<td>Zoltan Mann, Andreas Metzger</td>
</tr>
</tbody>
</table>
## Contributors

<table>
<thead>
<tr>
<th>Name</th>
<th>Organization</th>
<th>Sections</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nazila Golmohammadi</td>
<td>UDE</td>
<td>2</td>
</tr>
<tr>
<td>James Greig</td>
<td>OCC</td>
<td>2, 4</td>
</tr>
<tr>
<td>Zoltan Mann</td>
<td>UDE</td>
<td>1, 2, 5</td>
</tr>
<tr>
<td>Andreas Metzger</td>
<td>UDE</td>
<td>2</td>
</tr>
<tr>
<td>Stefan Schoenen</td>
<td>UDE</td>
<td>3</td>
</tr>
<tr>
<td>Mike Surridge</td>
<td>IT Innovation</td>
<td>4</td>
</tr>
<tr>
<td>Toby Wilkinson</td>
<td>IT Innovation</td>
<td>4</td>
</tr>
</tbody>
</table>
## Document History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>V0.1</td>
<td>December 2017</td>
<td>Initial version</td>
</tr>
<tr>
<td>V0.2</td>
<td>February 2018</td>
<td>Improvements based on discussions during the integration meeting in Essen</td>
</tr>
<tr>
<td>V0.3</td>
<td>March 2018</td>
<td>Description of the design process leading to the architecture</td>
</tr>
<tr>
<td>V1.0</td>
<td>April 2018</td>
<td>Finalization based on internal and cross-WP review</td>
</tr>
</tbody>
</table>
Contents

List of Figures ......................................................................................................................... 6
List of Tables ............................................................................................................................. 7
1 Introduction .......................................................................................................................... 8
2 Architecture .......................................................................................................................... 9
   2.1 Overview of Architecture Development Process ............................................................... 9
   2.2 Requirements Engineering ............................................................................................... 10
   2.3 Design ............................................................................................................................... 13
      2.3.1 High-level architecture and components ................................................................. 13
      2.3.2 Relation of RestAssured to the Context Entities ...................................................... 16
      2.3.3 Interface Descriptions ............................................................................................ 17
      2.3.4 Design constraints ................................................................................................. 18
   2.4 Validation ......................................................................................................................... 19
3 Testbed ................................................................................................................................ 23
   3.1 Infrastructure ................................................................................................................... 23
   3.2 Set-up ................................................................................................................................ 24
4 Methodology .......................................................................................................................... 25
   4.1 The Service Supply Chain ............................................................................................... 25
   4.2 Reusable Risk Models ...................................................................................................... 26
   4.3 Risk Model Assembly ...................................................................................................... 28
   4.4 Design-Time versus Run-Time Risk Evaluation ............................................................. 30
5 Conclusion ............................................................................................................................. 31
List of Figures

Figure 2.1: Development Process of RestAssured End-to-End Architecture .................................................. 10
Figure 2.2: i* Notation ...................................................................................................................................... 11
Figure 2.3: Simplified i* Goal Model including Data Protection Goals ............................................................. 12
Figure 2.4: Conceptual End-to-End Architecture for Data Protection .............................................................. 14
Figure 2.5: Satisfaction of Run-Time Data Protection Assurance by Access Control ........................................... 19
Figure 2.6: Satisfaction of Run-Time Data Protection Assurance by Access Control for Policy Enforcement .......... 20
Figure 2.7: Satisfaction of Run-Time Data Protection Assurance by Adaptation and Resolving Conflict with Cost Effective Decisions ...................................................................................... 20
Figure 2.8: Satisfaction of Run-Time Data Protection Assurance by Risk Assessment ........................................... 21
Figure 2.9: SCANT Architecture ..................................................................................................................... 22
Figure 3.1: The testbed at the UDE test lab ........................................................................................................ 23
Figure 3.2: The testbed architecture .................................................................................................................. 24
Figure 4.1: Information risk management standards ............................................................................................ 27
List of Tables

Table 2.1: Components of the RestAssured run-time system ........................................ 15
Table 2.2: Entities in the operational context of the RestAssured run-time system .......... 16
Table 2.3: Interfaces required from context entities .................................................. 17
Table 2.4: Interfaces required from RestAssured components .................................... 18
1 Introduction

This deliverable presents the first high-level architecture and methodology of RestAssured, thereby addressing over-arching technical and methodological considerations of the project. This deliverable forms a foundation for the other deliverables that describe the details of the technical work packages (WP4-WP7) and the use cases (WP8). Moreover, this deliverable acts as a blueprint for the other technical work packages, ensuring consistency and compatibility of the components developed within the individual work packages and thus ensuring that the interplay of the individual components adds up to an integrated solution that brings value to the affected stakeholders.

This deliverable is considerably different from its predecessor D3.1, which presented an initial description of the RestAssured high-level architecture. D3.1 was based on the consortium’s understanding of the architectural needs and possible solutions at month 4 of the project which were captured by several different architectural views (data flow view, risk analysis view, adaptation view, component view) in D3.1. Since then, the consortium has significantly developed, extended, and refined its vision of the RestAssured solution, leading to new insights ultimately converging on an agreed high-level architecture described in terms of the RestAssured run-time components.

This deliverable, D3.2, is organized as follows. Chapter 2 describes the overall high level architecture of RestAssured. This architecture governs the interplay of the RestAssured run-time components with each other and with the environment. The architecture describes the components and their interfaces on a logical level – more details about the components and the interfaces, particularly with regard to the used technologies – are provided in the respective deliverables of the individual work packages.

Chapter 3 describes the RestAssured testbed: the physical and virtual environment used for deploying, integrating, and testing the RestAssured technical components. The testbed is also used for demonstration purposes.

Chapter 4 describes the RestAssured methodology, explaining a continuous data protection risk assessment process, which starts at design time and employs the RestAssured architecture for run-time concerns. The methodology starts from the business perspective, and the need under the GDPR for (continuous) risk assessment by service operators, and the consequent issues this raises for service supply chains. A solution based upon reusable risk models is outlined, and the relationship between design-time and run-time risk assessment discussed.

Finally, Chapter 5 sums up the main points of the document and discusses conclusions as well as directions for future work.
2 Architecture

This chapter describes the RestAssured architecture. RestAssured encompasses a wide-ranging set of different solutions, techniques, technologies, methodologies, and tools. It is therefore difficult to present RestAssured as a whole in a concise and coherent manner. To overcome this difficulty, the deliverable D3.1 used multiple architectural views to describe the RestAssured architecture. However, as RestAssured has evolved and the technical integration of its components has become increasingly important, the need arose to have a single architectural picture governing the interplay of the RestAssured components with each other and with the environment.

To be able to integrate all RestAssured components in a single architectural diagram in spite of the differences between the components in terms of position in the technology stack, level of abstraction, and methodological role, some compromises and restrictions or limitations had to be introduced:

- The architecture focuses on a conceptual decomposition into logical components and the conceptual interplay among these components. Technology aspects and deployment aspects are not included in the architecture. (Such considerations are described in the respective work package deliverables as necessary.)
- The architecture presents the interplay of components during run time. Activities performed and tools used at design time are not captured by the architecture. (See Chapter 4 for more details about design-time activities.)
- The architecture presents the high-level components of RestAssured. Further decomposition within the high-level components is not discussed in this deliverable, but rather in the deliverables of the individual technical work packages WP4-WP7.

In this chapter, we describe the design considerations to devise an end-to-end (E2E) architecture for run-time data protection, i.e., the RestAssured run-time system. We start with an overview of the process that we followed to develop the E2E architecture of RestAssured for run-time data protection in Section 2.1. Section 2.2 explains the requirements engineering phase of the process and the major outcomes of this phase. Then, we present in Section 2.3 the conceptual E2E architecture for run-time data protection as a result of the design phase, including the components of the architecture, as well as the interfaces among the individual components and between the RestAssured run-time system and its context. Section 2.4 details the validation phase, and describes how the presented architecture fulfills the identified requirements, which includes the application of the architecture to the OCC CARE use case.

2.1 Overview of Architecture Development Process

We first give an overview of the process to develop and validate the E2E architecture for run-time data protection, and the major artefacts resulting from each phase of the process. We followed a systematic and iterative process with three phases, as shown in Figure 2.1.

The first phase is concerned with requirements engineering activities [?], which include context analysis and identifying requirements using goal- and scenario-based approaches (1a and 1b respectively in Figure 2.1). Context analysis aims to identify the actors involved and the initial context information. In particular, we identify the different actors involved in a cloud environment in their roles related to data protection. For the identification of requirements we use a goal modelling approach. This step is concerned with setting up the goal model by capturing the major intentions of the involved actors. We start with high-level goals, from which we derive the actual requirements. We use the i* [?] notation for this purpose.

The second phase is the actual design of the conceptual architecture that addresses the requirements and satisfies the identified goals. This involves a decomposition into a set of components, definition of the
The third phase is the validation of the developed E2E architecture. We do this using scenario-based goal satisfaction analysis (step 3a in Figure 2.1), i.e., using a scenario for each goal of the goal model (regarding data protection) that demonstrates the satisfaction of the goal. Complementing this form of validation, we instantiate the E2E architecture in a commercial case study, namely the OCC CARE use case (step 3b).

Feedback gathered during the validation phase is fed back to the requirements engineering and design phases as indicated.

The whole design and validation process was carried out using feedback from and discussions with all project partners.

2.2 Requirements Engineering

To determine the roles and requirements related to data protection, we consider two key formal documents that define data protection principles, and which help identifying the related actors and entities. On the one hand, we consider the privacy framework as defined in the international standard ISO/IEC 29100 [1]. On the other hand, we consider the EU GDPR [2], which is the currently binding legal framework in the EU member states.

In the following, we enumerate the identified roles and actors and the identified requirements. We structure the run-time data protection requirements into an i* goal model and address this goal model in the E2E architecture introduced in Section 2.3.

1a) Context Analysis

ISO/IEC 29100 and the EU GDPR partly overlap and do not use the same terms consistently. To avoid ambiguities, we use the following definitions for the key entities and most important actors relevant for our E2E architecture:

- **Data Subject** is an identifiable, natural person, who can be identified directly or indirectly, in particular by reference to an identifier such as a name, identification number, location data, online identifier or to one or more factors specific to the physical, physiological, genetic, mental, economic, cultural or social identity of that natural person [3]. This term is called Personally Identifiable Information (PII) principal in ISO/IEC 29100. Here, a data subject represents a person about whom data are to
be stored and/or processed in the cloud. The data subject has, with respect to the personal data about them, the rights stipulated by the GDPR. We consider the end-user as a data subject.

- **Data Controller** is the natural or legal person, public authority, agency or other body which, alone or jointly with others, determines the purposes and means of the processing of personal data [?]. This term is called PII controller in ISO/IEC 29100. Here, a data controller represents a legal entity providing a cloud service which stores and/or processes personal data. The data controller has, with respect to the stored/processed personal data, the obligations stipulated by the GDPR. We consider the cloud provider at the first interaction point with the end-user as the data controller.

- **Sensitive Data**: The focus of GDPR is on personal data. Personal data means any information relating to an identified or identifiable natural person (data subject) [?]. This term is called PII in ISO/IEC 29100. We consider sensitive data as the data stored in the cloud that needs to be protected in line with the data protection preferences of the affected data subject. In particular, we extend the notion of sensitive data to also include confidential business data.

- **Application** represents a cloud application that works with sensitive data. The data controller is responsible for the legal and compliant operation of the application.

- **Infrastructure** represents the physical and virtual infrastructure that hosts the application as well as the sensitive data.

1b) Requirements Identification using Goal and Scenario Modelling

We use the i* notation for both the early and late phases of the requirements engineering process. During the early requirements phase, the i* framework is used to model the context of the system-to-be. It allows requirements engineers to represent the actors of the system, their objectives, and their relationships. The i* models developed at this stage help in understanding why an E2E run-time data protection system is needed. During the validation phase, the i* models are used to analyse the developed E2E architecture and its components and evaluate them based on how well they meet the established goals and requirements. The basic elements of the i* modeling notation are shown in Figure 2.2.

Our aim is to provide an E2E architecture as a solution to specific technical concerns of data protection among different actors in the cloud. We capture these concerns in goals, and refine them into the requirements that the E2E architecture should fulfill.

In Figure 2.3, an i* goal model for the data controller is depicted. The overall goal of the data controller is “Making profit”. This goal is decomposed into several subgoals: “Providing cloud services”, “Avoiding
penalties”, “Gaining reputation” (which is a soft goal, the fulfillment of which does not have a clear-cut specification) and “Optimising resource allocation”.

Providing a cloud service needs to respect data protection, which is challenging and expensive for data controllers. A data controller is usually responsible for assuring conformance to data protection regulations such as the GDPR, while providing the cloud services. “Conforming to regulations” can satisfy the goal of “Avoiding penalties” while having positive influences on the soft goal of “Providing data protection respecting services” as well as having “Satisfied service users”.

The data controller can achieve the goal “Conforming to regulations” by “Practicing privacy-by-design” methodologies and by providing “Run-time data protection assurance”. Our focus is on run-time aspects, and thus we decompose the “Run-time data protection assurance” goal into three tasks: (i) “Risk assessment” based on the resource of a “Run-time Model”, (ii) “Adaptation” upon “Detection of violation with respect to data protection”, which necessitates “Monitoring Capabilities” for observation of the cloud architecture and identification of changes of cloud service deployments and (iii) “Access control”, which necessitates to “Send data protection related notifications”, “Request consent” and “Data protection respecting disclosures”.

To allow for “Optimising resource allocation”, “Making cost effective decisions” is necessary, which can have a negative influence on “Providing data protection respecting services”. Hence, some conflict resolution needs to be devised.

Figure 2.3: Simplified i* Goal Model including Data Protection Goals.
Note that empowering the data subjects in taking control on how their data will be handled in the cloud environment and in specifying their data protection preferences is included in the “Access control” task.

Based on the goal model presented in Figure 2.3, we derived the following requirements that the E2E architecture needs to satisfy. The requirements of different actors in the context can be summarised as follows:

R1 Data subjects should be able to register to the E2E run-time data protection system to specify and update their data protection preferences.

R2 Accesses to sensitive data of data subjects are only permitted if allowed by the relevant data protection policies.

R3 Data controllers should be able to register to the E2E run-time data protection system to specify contracts of offered services.

R4 Applications should be able to request access to sensitive data.

R5 The application’s accesses to sensitive data should always comply with the data protection policies.

R6 Data controllers should be able to monitor applications, the cloud infrastructure and changes in data protection policies.

R7 Violations against data protection policies should be identified.

R8 Data controllers should be supported in performing adaptations on applications and the cloud infrastructure.

R9 Data controllers should be supported in identifying risks with respect to data protection policies, thereby facilitating proactive adaptations.

2.3 Design

In this section, we present our conceptual end-to-end architecture and the description of its components. In addition, we explain their relation to the context entities, provide a high-level description of component interfaces, and state important design constraints considered during the design phase.

2.3.1 High-level architecture and components

Figure 2.4 presents an overview of the conceptual E2E architecture for run-time data protection. The thick green frame is the boundary between an application- and infrastructure-agnostic data protection service (the “system”) created by RestAssured and the set of entities that interact with the run-time data protection system but are beyond the control of our approach (the “context”).

For the right interpretation of the overview diagram, the following assumptions have to be taken into consideration:

- The diagram only presents logical components and their logical relations. The deployment of the components, also including considerations regarding the number of instances of each component, colocation or integration of multiple components, as well as centralisation/decentralisation, distribution of the components, is neither covered nor prescribed by the architecture diagram.

- To simplify the representation, only a single application and a single data controller are shown in the diagram. In practice, multiple applications and data controllers can be handled by the same run-time data protection service instance.
- The diagram only shows relations among components and relations between a component and a context entity. Relations among context entities are not shown.

As depicted in Figure 2.4, the E2E architecture of the run-time data protection system consists of five functional components (represented by rectangles) and a commonly used data store for the run-time model (represented by an oval). These are described as follows (see also Table 2.1 for an overview):

- The Run-time Model is a model of all relevant assets and their relationships both within the system and in its context. The model is kept up-to-date using monitoring. The information in the model is used by multiple components to reason about the current situation, the associated risks of data protection violation or other requirement violations. This component addresses the requirements R5, R6, R7, R8, and R9.

- The Data Gatekeeper manages the data protection policies and service contracts governing the data life-cycle. Data protection policies are specified either by data subjects to capture their individual privacy preferences, or by superordinate actors to specify general rules of data protection (legislation, company policies etc.). Service contracts are established with service providers in their role as data controller, defining what operations their service performs on which kinds of data. The Data Gatekeeper is responsible for deciding, based on the available policies and contracts, which operations are allowed on which piece of data. This component addresses the requirements R1, R2, and R3.

- The Data Access Protection component is responsible for ensuring that data accesses are secure and conform to the relevant policies. To ensure data confidentiality and integrity, the Data Access Protection component applies secure enclaves and cryptographic techniques: the data are stored in encrypted form and their decryption takes place in a secure environment, either within a secure hardware enclave or on a trusted machine (e.g., in a private data center). This way, unauthorized parties cannot get access to the cleartext. Moreover, the Data Access Protection component is involved in access control to enforce the compliance with the specified data protection policies. The “Access control”
Table 2.1: Components of the RestAssured run-time system

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
<th>Work package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data gatekeeper</td>
<td>Manages the policies and contracts governing the data life-cycle; decides which data accesses are allowed</td>
<td>WP6</td>
</tr>
<tr>
<td>Data access protection</td>
<td>Ensures – using secure enclaves and cryptographic techniques – that data accesses are secure and conform to the relevant policies</td>
<td>WP4</td>
</tr>
<tr>
<td>Adaptation</td>
<td>Ensures the satisfaction of requirements in spite of changes, by monitoring the system and its environment and making adaptations as necessary</td>
<td>WP5</td>
</tr>
<tr>
<td>Risk assessment</td>
<td>Continuously assesses risks associated with the current system setup as well as with planned adaptations</td>
<td>WP7</td>
</tr>
<tr>
<td>Logging</td>
<td>Non-repudiable logging of events</td>
<td>3rd-party tool</td>
</tr>
<tr>
<td>Run-time model</td>
<td>A model of the relevant assets and their relationships, kept alive using monitoring</td>
<td>WP5</td>
</tr>
</tbody>
</table>

from the goal model is addressed by the Data Gatekeeper and Data Access Protection components jointly. The Data Access Protection addresses the requirements R2, R4, and R5.

- Adaptation is responsible for the satisfaction of requirements in the presence of run-time changes. To this end, the Adaptation component continuously monitors the system and its environment. If a change is detected, its impact on data protection and other quality attributes is analyzed. If an actual or imminent problem is identified, Adaptation devises a plan to adapt the system such that the problem is avoided or mitigated. Finally, the adaptation is carried out by re-configuring the appropriate component or context entity. This component addresses the requirements R6, R7, and R8.

- Risk Assessment is responsible for continuous run-time assessment of risks. On the one hand, it assesses risks associated with the current system setup, and triggers the Adaptation component if the risk level is too high. On the other hand, it assesses the risk impact of planned adaptations to ensure that any changes proposed by adaptation will be compliant with the available policies and do not introduce unacceptable risks of data protection violation. This component addresses the requirements R7 and R9.

- The Logging component enables the provision of an audit trail. For this purpose, all components log all of their activities that are relevant to data protection to a central log server providing a non-repudiable logging service.

Each component except for Logging is developed within one of the technical work packages WP4-WP7 (see last column in Table 2.1). The detailed description of the components can be found in the deliverables of the respective work packages.

The main data and control flows among the RestAssured components can be summarized as follows:

- The Data gatekeeper and Data access protection components cooperate to ensure that an application can only access data that application is allowed to access, based on the data protection policies specified by the individual Data Subjects. This includes requests to create, read, write, or delete data.

  - An important special case is when the application wants to create a new data set (e.g., a new table in a relational database). In that case, the application also specifies which attributes are to be considered sensitive – this information is forwarded by the Data access protection component to the Data gatekeeper.
• If a Data Subject changes their data protection policy, the Data gatekeeper informs the Application about this change, because the change might have further application-specific consequences (e.g., in the Pay-As-You-Drive use case, a change in data protection policies may impact the Data Subject’s insurance plan).

• Adaptation uses monitoring information – obtained from both context entities and RestAssured components – to keep the Run-time model accurate and up-to-date.

• The Risk assessment component analyzes the Run-time model and if it detects a situation in which the risk of data protection violation is too high, the Risk assessment component triggers the Adaptation component.

• The Adaptation component proposes adaptations to Risk assessment. If an adaptation is approved by the Risk assessment component, the Adaptation component executes the adaptation through the appropriate adaptation interfaces.

The interfaces that allow these data and control flows are described in Section 2.3.3.

2.3.2 Relation of RestAssured to the Context Entities

The operational context, or simply context, of the RestAssured run-time system consists of all the entities that interact with the RestAssured run-time system but do not belong to the system itself. These are the entities in Figure 2.4 that are outside the thick green box.

<table>
<thead>
<tr>
<th>Entity</th>
<th>Description</th>
<th>Connection to RestAssured</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Subject</td>
<td>Represents a person about whom data are to be stored and/or processed in the cloud. The Data Subject has, with respect to the personal data about them, the rights stipulated by the GDPR</td>
<td>RestAssured makes it possible to capture and enforce the data protection requirements of the Data Subjects</td>
</tr>
<tr>
<td>Data Controller</td>
<td>Represents a legal entity providing a cloud service which stores and/or processes personal data. The Data Controller has, with respect to the stored/processed personal data, the obligations stipulated by the GDPR</td>
<td>RestAssured helps the Data Controller to be compliant with the GDPR by enforcing the relevant data protection policies</td>
</tr>
<tr>
<td>Sensitive data store</td>
<td>Data stored in the cloud that needs to be protected in line with the affected Data Subjects’ data protection preferences</td>
<td>RestAssured ensures that accesses to sensitive data are only permitted if the relevant data protection policies allow them</td>
</tr>
<tr>
<td>Application</td>
<td>Represents a cloud application that works with sensitive data. The Data Controller is responsible for the legal and compliant operation of the Application</td>
<td>RestAssured ensures that the Application’s accesses to the sensitive data comply with the data protection policies</td>
</tr>
<tr>
<td>Cloud infrastructure</td>
<td>Represents the physical and virtual infrastructure that hosts the Application as well as the Sensitive data store</td>
<td>RestAssured takes the characteristics, restrictions, and possibilities of the Cloud infrastructure into account when making decisions relating to data protection</td>
</tr>
</tbody>
</table>
Table 2.2 describes the context entities and their relation to the RestAssured run-time system. The data and control flows between the RestAssured run-time system and its context can be summarized as follows:

- The actors – Data Subject and Data Controller – register with the RestAssured run-time system to specify their data protection preferences respectively their offered services.
- Access requests of applications to sensitive data (including requests to create, read, write, or delete data) are mediated by the RestAssured run-time system, ensuring that access to sensitive data conforms to the relevant data protection policies.
- Applications and the cloud infrastructure are monitored by the RestAssured run-time system. If necessary, the RestAssured run-time system can also perform adaptations on applications and the cloud infrastructure.

The interfaces that allow these data and control flows are described in Section 2.3.3.

### 2.3.3 Interface Descriptions

In this section, we describe for each of the RestAssured components and context entities which interfaces they require and from which component or from which context entity. The details on how the components provide the required interfaces are described in the respective work package deliverables.

#### Table 2.3: Interfaces required from context entities

<table>
<thead>
<tr>
<th>Required by</th>
<th>Provided by</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Data Subject</td>
</tr>
<tr>
<td>Data gatekeeper</td>
<td>–</td>
</tr>
<tr>
<td>Data access protection</td>
<td>–</td>
</tr>
<tr>
<td>Adaptation</td>
<td>–</td>
</tr>
<tr>
<td>Risk assessment</td>
<td>–</td>
</tr>
<tr>
<td>Run-time model</td>
<td>–</td>
</tr>
</tbody>
</table>

The interfaces are described in tabular form. To enhance readability, the information is split into two tables: Table 2.3 presents the interfaces required from context entities, while Table 2.4 presents the interfaces required from the RestAssured components. It should be noted that Table 2.3 is shorter than Table 2.4 because interfaces among context entities are not considered.

Both Table 2.3 and Table 2.4 are structured as follows: the rows correspond to entities/components that require an interface from some other entities/components; the first cell of each row names the entity or component requiring the interface. The columns (except for the first column) correspond to the entities/components that provide the interfaces. For example, the cell in Table 2.4 in row “Data Subject” and column “Data gatekeeper” describes an interface provided by the Data gatekeeper and used by Data Subjects. It should be noted that who provides and who requires an interface is independent of the direction of the data flow.
Table 2.4: Interfaces required from RestAssured components

<table>
<thead>
<tr>
<th>Required by</th>
<th>Provided by</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Data gate-keeper</td>
</tr>
<tr>
<td>Data Subject</td>
<td>Registration of data protection policies</td>
</tr>
<tr>
<td>Data Controller</td>
<td>Registration of offered service</td>
</tr>
<tr>
<td>Sensitive data store</td>
<td>–</td>
</tr>
<tr>
<td>Application</td>
<td>–</td>
</tr>
<tr>
<td>Cloud infrastructure</td>
<td>–</td>
</tr>
<tr>
<td>Data gate-keeper</td>
<td>–</td>
</tr>
<tr>
<td>Data access protection</td>
<td>Set of data subjects allowing a given query</td>
</tr>
<tr>
<td>Adaptation</td>
<td>–</td>
</tr>
<tr>
<td>Risk assessment</td>
<td>Information on policies and service contracts</td>
</tr>
<tr>
<td>Run-time model</td>
<td>–</td>
</tr>
</tbody>
</table>

2.3.4 Design constraints

This section describes essential design constraints we took into account when developing the RestAssured architecture. These concern overarching design decisions that are not specific to a component but rather relate to the RestAssured run-time system as a whole. The following constraints were identified:

- A fundamental assumption underlying the interfaces between the RestAssured run-time system and its context is that the context entities trust the RestAssured run-time system. The trustworthiness of the RestAssured run-time system is ensured using appropriate technical solutions – e.g., using secure hardware enclaves – as described in the respective work package deliverables. Since the context entities trust the RestAssured run-time system, the context entities can be expected to provide the necessary interfaces, as described in Table 2.3.
• For the cooperation of the RestAssured components it is important to uniquely identify objects of interest, using the same identifier to refer to an object across the different RestAssured components. To ensure such coherent identification of objects, identifiers should be generated for all objects of interest by a designated component. Because of its central role in providing information about the cloud system, the Run-time model component acts as the central source of object identifiers.

• To warrant maximum flexibility in the development and evolution of the RestAssured components, the components should be loosely coupled. The interaction among the components should be done only through the well-defined interfaces, as described in Section 2.3.3. Appropriate interconnection technologies should be used to minimize technical dependencies among the components (e.g., using REST services).

2.4 Validation

In this section, we validate our E2E architecture by using a scenario-based goal satisfaction validation approach on the one hand, and a case study from one of the commercial partners of RestAssured on the other hand.

3a) Scenario-based Validation

We use a scenario-based approach to examine the satisfaction of the major goals related to run-time data protection assurance. Figures 2.5–2.8 depict some goal satisfaction scenarios in the form of UML sequence diagrams. We focus on the task decomposition of the “Run-time data protection assurance” goal.

Figure 2.5 shows the scenario that satisfies “Run-time data protection assurance” by the task “Access control”. “Access control” is addressed by the components Data Gatekeeper and Data Access Protection. Data subjects provide their data protection preferences related to their sensitive data. Data controllers provide the contract information of their offered services. For instance, the contract information includes information about what data is used and how it is used. This information is provided to the Data Gatekeeper. Once an Application requests access to the sensitive data of data subjects from Data Access Protection, the Data Access Protection sends this request to the Data Gatekeeper. The Data Gatekeeper allows or restricts the data accesses. In any case, the Data Gatekeeper ensures the enforcement of the data protection policies (based on the preferences of the data subjects and the data protection regulations). The enforcement of the data protection policies is depicted in Figure 2.6.
Figures 2.7 and 2.8 show the scenarios satisfying “Run-time data protection assurance” by the tasks “Risk assessment” and “Adaptation”, based on the “Run-time Model” as a resource.

Figure 2.7 depicts also the conflict resolution between making cost effective decisions and ensuring data protection (cf. Figure 2.3). Once the Adaptation component comes up with an adaptation plan (e.g., migrations among physical machines or data centers for the sake of cost and energy efficiency), this plan is assessed by the Risk Assessment component. Risk Assessment uses the current configuration of the system from the run-time model and information about service contracts and data protection policies to evaluate the proposed adaptation plan. Based on the evaluated risk, adaptation can be performed or is denied.

Figure 2.8 shows how the Risk Assessment component can also trigger the Adaptation component to adapt the system configuration once it determines that risks of data protection violation are too high in the current system configuration. This may lead for instance to restrictions of data accesses.

3b) Case Study

In this section, we describe how the proposed architecture is used in a real-world, commercial application that handles personal data in the social care domain, namely the OCC CARE use case of the project.
Ami, developed and operated by Oxford Computer Consultants, is an online service in the United Kingdom which connects (i) lonely or socially isolated people who need help and (ii) volunteers offering help. Matching volunteers to people needing care is done based on information such as place where the person lives and their needs. These pieces of information are displayed only in obfuscated form, so as to preserve the users’ privacy.

The information about people with loneliness and related needs is valuable to local authorities, who have responsibility for supplying social care to persons in need within their areas. SCANT is a tool to make use of this information to assist the local authorities in provisioning of social care and identify unmet needs, whilst also preserving the privacy and safety of the potentially vulnerable Ami users. For instance, local authorities can query with SCANT the number of Ami users with particular needs in a broad geographical region – however, individual Ami users who did not consent to the disclosure of their data must remain anonymous to the local authorities.

Figure 2.9 shows the SCANT architecture. As can be seen, it is an instantiation of the conceptual architecture presented in Figure 2.4, with the following refinements:

- The registration of data subjects takes place with a special user interface (Ami/SCANT registration tool), which forwards the information about user consent to the Data Gatekeeper.
- Currently, the SCANT application is neither monitored nor adapted. However, adaptations of the Data Access Protection component also indirectly impact the way SCANT can access data.
- For storing sensitive user data, the Opaque secure data analytics platform is used, which provides additional guarantees about data confidentiality and integrity [2]. Opaque uses secure hardware enclaves provided by Intel’s SGX technology.

By using the proposed architecture, data protection policies of the Ami users can be captured and enforced throughout the data lifecycle. The stored sensitive data are protected against unauthorized access. Queries

---

1https://www.withami.co.uk/
of local authorities are modified automatically on the fly so that the data of Ami users who did not consent to the analytical use of their data are excluded from the results. This guarantees that local authorities never get access to data of Ami users who did not consent to this. On the other hand, local authorities can still work with data of Ami users who did consent to the disclosure of their data as well as with aggregated data of Ami users who consented to this. This automatic fine-grained access control is a major advantage of the architecture. Furthermore, through continuous monitoring, risk assessment, and adaptation, also changes to the underlying infrastructure can be handled in a way that is transparent to the application. For instance, it is possible to switch between multiple Opaque nodes on the fly provided that they offer similar protection levels, as determined by run-time risk assessment.

Figure 2.9: SCANT Architecture.
3 Testbed

In this chapter we describe the testbed at month 15 of the project. The purpose of the testbed is to allow all partners to deploy, integrate, and test their software components on a common infrastructure. Furthermore, the testbed is used for demonstration purposes. We describe the hardware and software infrastructure of the testbed and how it is set up for the use in RestAssured.

3.1 Infrastructure

The testbed consists of six Fujitsu Celsius w550 workstations, each equipped with

- Intel Xeon E3-1275v5 processor with 3.6 GHz clock frequency
- 16 GB DDR4 memory
- 1000 GB SATA III HDD and 512 GB SATA III SSD

For easier understanding, we named the testbed workstations TB1, TB2, TB3, TB4, TB5 and TB6. All six are connected to a HP 2530-8G switch over which every incoming connection runs. In Figure 3.1 the six workstations at the UDE test lab can be seen.

![Figure 3.1: The testbed at the UDE test lab](image)

All six workstations’ BIOSes run Intel SGX in “enabled mode.” The operating system on each workstation is Ubuntu 16.04 with kernel version 4.10.0-42-generic. Intel SGX software is installed on all workstations as well. Originally, Intel SGX 1.8 was installed on each workstation. Due to an update to the Query Gateway within the Data Access Protection component, the current SGX version on TB2 and TB3 has been upgraded to 2.0.

The six testbed workstations are accessible via Port 22 (SSH). They are all connected to a switch which is connected to the UDE network. To prevent unauthorized access to any workstation, each workstation is secured by the UDE firewall. The network architecture is illustrated in Figure 3.2.

---

3.2 Set-up

Testbed administration is done by UDE. To ensure controlled access to the testbed resources, each employee of the partner organizations involved in the project and wanting to use the testbed was given a user account. The accounts are created and managed by UDE. Each partner deploys their software to the testbed and manages it on their own. Ports are managed by UDE. Version control for the source code of the software is not provided on the testbed but on github.

The currently opened ports for incoming connections for each workstation are:

- Ports 80 and 8080 are used to test web applications as ports for HTTP
- Port 443 is used to establish secure web access for some components
- Port 8282 was opened to allow database access for the central runtime model
- Port 9000 is used for the query gateway of the Data Access Protection component
4 Methodology

The starting point for the overall RestAssured methodology is to view things from a business perspective, rather than a technical one.

Under the GDPR, the obligation to protect privacy is placed on the Data Controller, typically the service operator, rather than on the suppliers of any technical solutions that are used to deliver that service. Alongside specific obligations to allow data subjects to give or withdraw consent for data processing and view and correct data, the Data Controller has to carry out a risk analysis, and use security measures that appropriately manage that risk. The GDPR also explicitly states that risks must be assessed not only at the time the data collection and processing arrangements are defined, but also continuously during their operation. Data Processors also carry an obligation to assess and address risks under the GDPR – they cannot simply do as the Data Controller specifies. Thus service operators will normally be responsible for performing (continuous) risk assessment required under the GDPR.

When using non-personal but commercially sensitive data, the GDPR itself does not apply. However, the same principles apply when protecting data of any kind. While there may be no obligation on a service operator to analyse, assess and address risks, it remains the best approach available to assure the security of sensitive data. It is also consistent with international standards on information security risk management.

4.1 The Service Supply Chain

Service operators often use external suppliers for software and other technologies, and as a result, they are often not in a position to perform such analysis as they have insufficient technical knowledge of the technologies used to deliver the service. The service operator must therefore rely on their suppliers to help conduct any risk assessment.

The current state of the art is for software suppliers to use penetration tests to verify the security of their software against certain types of attacks. This provides a service operator with some assurance that the software they use is secure against some attacks. However, this approach leaves many open questions including:

- How should such tests be defined, and by whom?
- How should such tests be carried out, and by whom?
- How should the results be interpreted in the context of a risk assessment by a service operator?

The service operator is responsible for the risk assessment, but they may know very little about the technologies employed, and therefore may neither be in a position to define what the tests should be, nor to assess the results of those tests. Equally, the technology supplier may know little about the context in which the technology is employed, and therefore may also not be in a position to know which types of threats pose the greatest risk, and which are relatively unimportant.

The requirement to perform continuous risk assessment adds further complications to existing processes. Penetration testing is labour intensive and requires considerable technical expertise, and may not be available on a continuous basis. Periodic penetration tests could be performed, but what is an appropriate interval, and are there limitations on how intrusive such tests can be on a live system?

In addition to software, service operators often use third party (Cloud) infrastructure to host their services. This is of course the main focus for the RestAssured project. The best methodology currently available for dealing with complications arising from the use of cloud services is the guidelines published by the Cloud Security Alliance [?]. This states that responsibility for assessing risks at the level of an enterprise remains with the enterprise, i.e. one cannot outsource this responsibility to a cloud service provider. Partly as a
consequence, the guidelines place a great emphasis on the assessment of risks and countermeasures by cloud service customers. They are expected to request or acquire documentation, review the cloud service provider’s security program and documentation along with any legal, regulatory, contractual and jurisdictional requirements (on both sides), and then evaluate the contract’s service in the context of information assets. One should also separately evaluate the provider’s financial stability, reputation and so on.

In many ways, the approach for analysing risks involving cloud service providers is worse than for service software providers. Service software does play a direct role in the security (or otherwise) of a service-based application, and penetration testing is directly relevant, although not easy to relate to specific requirements coming from risk analysis. With cloud services, the state of the art involves analysing risks more or less independently of what the cloud service provider offers.

A smarter, more scalable, and more business friendly solution is required. Such a solution would employ a methodology where technology suppliers, cloud service providers and service operators can cooperate within the supply chain to analyse risks. The key elements of this approach must be:

- Service technology developers provide reusable risk modelling components, representing their software including its security properties and the context in which it is intended to operate, backed up by documentation from relevant penetration tests.

- Cloud service providers provide further risk modelling components representing the assets they provide (infrastructure assets, platforms or software as a service), including security properties backed up by relevant documentation as required.

- Data controllers assemble these reusable contributions to create a model of their overall process, within which risks can be identified and assessed.

To implement such a sophisticated approach poses many challenges, including many that were not foreseen in the RestAssured proposal. The methodology described briefly in RestAssured deliverable D3.1 is not fully consistent with such a process, the need for which emerged during the development of the first prototypes. Our focus at this stage has been to understand how to handle the challenges of automated design-time risk assessment, and extending this to run-time risk assessment. Current prototypes allow a user to create a model of the system as the starting point, but we have not started to address the fact that in practice, the model should incorporate knowledge coming from several stakeholders in the supply chain.

The remaining methodological challenges will be addressed in the second half of the project. Note however that some steps (e.g. any changes in penetration testing methods) lay beyond the scope of RestAssured. It is therefore unlikely that the methodology as a whole can be realised within the lifetime of the project. RestAssured can show how some of the key challenges of risk analysis in multi-stakeholder systems can be addressed, before and during run-time, and how risks can then be addressed using sticky policies, advanced encryption and trusted computing mechanisms.

4.2 Reusable Risk Models

In many ways the problem we are addressing is not a new one, but regulations like the GDPR have brought the need for risk assessment into ever sharper focus.

Figure 4.1 shows the relationship between existing standards for risk management in information systems, where risk assessment falls under ISO 27005, penetration testing under ISO 15408, and ISO 27001 provides the overall framework.

Looking at Figure 4.1 the direction of the flow of control is from ISO 27005 (risk assessment) to ISO 15408 (penetration testing). The risk assessment defines what controls need to be in place, and penetration testing determines whether they are, and whether they are effective.
In many cases though, we want to reverse this hierarchy. A technology supplier will typically supply services to many service operators, but it is the service operators who are responsible for the risk analysis and the consequent flow of security control requirements to the technology suppliers. The service providers therefore determine what penetration testing needs to be performed, and this flows down to the technology suppliers. The technology suppliers may therefore receive many similar (but not necessarily identical) sets of requirements from their customers. From the perspective of a technology supplier, it makes more sense to test their products or services once, and then flow that information back to their customers in a way that the service operators can easily consume for their own risk assessments.

The ideal solution to the problem caused by the sequential relationship between ISO 27005 and ISO 15408 is to construct design-time risk models from reusable components. Technology suppliers would provide risk models for their components, and a service operator would then integrate these into a combined model for their service. The resultant composite design-time risk model would form the basis of the design-time risk assessment, and when fused with the run-time system model, the continuous run-time risk assessment of the deployed service.

To do this, it will be necessary to agree to some constraints on the context within which reusable components can be combined. One issue is the need to identify a common set of stakeholder roles, and their relationships to different types of components. This can be addressed in RestAssured by using the Cloud System Analysis Pattern (CSAP) methodology, using patterns to characterise these roles and relationships and provide a starting point for different stakeholders to produce their contributions. A key point is that these patterns should refer to roles, not to individual (and specific) stakeholders. These roles must then form part of the common taxonomy used by all contributors to risk models.

Service software developers could then contribute by constructing small system models that show how their components interact with other assets in the system. Specifically, which stakeholder role(s) operate the supplied components, how these relate to other components including other services and also data, and what relationships those other components have to stakeholder roles. The small system models may elaborate on the relationships between other service components and stakeholders, if these are assumed to exist by the
software developer. The principle should be to include only relationships that are assumed to form part of the context for the supplied software component. For example, if a service developer assumes that data is provided and consumed by the same stakeholder (implying that the service does not share data with any third party), that should be included in their context model. The same approach should also be used by Cloud service providers who offer Software as a Service (SaaS), possibly building on a model provided to them by the original software developer.

For Cloud service providers who supply platform or infrastructure as a service (PaaS or IaaS), the situation is slightly different. With these types of services, there is no fixed relationship to many stakeholders that interact with or are affected by services running on their platforms or infrastructure. In the case of IaaS, there is an obvious solution, however. The Cloud service provider is offering a host (or multiple hosts) on which services can be deployed. Hosts are considered to be assets within the risk modelling methodology described in D7.1, so IaaS providers (at least) can use the same approach. The assets they provide (virtual or physical hosts) can be incorporated into a small system model. In this case the direct relationships will be to the Cloud service provider itself, and to those stakeholder(s) who administer the provided hosts. These are unlikely to vary from the CSAP pattern, and involve only the administration interface and the security measures provided there and at the provisioned hosts. For example, if a Cloud service provider offers hosts with built-in security patching, or with hardware-enabled secure enclaves, those features should appear in the model.

The current RestAssured risk modelling ontology does not yet support a full range of risk mitigation features that may be provided by Cloud service providers. For now, we only address a subset of IaaS measures. This is something that can be addressed later in the project. The focus for this, e.g. the balance between a deep analysis of IaaS versus a less deep coverage of both IaaS and PaaS, will depend on the validation case study requirements and experiences from the initial validation experiments.

### 4.3 Risk Model Assembly

The final step is then for the service operator (e.g. Data Controller) to assemble an overall design-time model of their processing chain, including the cloud deployment and the relationships to other stakeholders. The contributions from software developers will need to be integrated into the design time model, because the software is fixed prior to deployment. The contributions from Cloud service providers become relevant only when considering which Cloud services to use.

The overall sequence is therefore expected to be as follows:

1. The CSAP tool from RestAssured will be used to create an initial template, defining the basic types of components and stakeholders involved in a typical system. This model should cover indirect stakeholders (e.g. regulators) as well as those directly engaged in producing, operating or using the system.

2. Cloud service providers will create small models describing the security measures they provide w.r.t. the provisioned assets and relationships to themselves and their customers. These models may be created using the System Security Modeller (SSM) tool from RestAssured, by loading the relevant template and annotating assets to indicate the available security measures. For PaaS it may be necessary to use a much richer model incorporating a collection of platform services, so at this stage we envisage focusing on IaaS only.

3. Service software developers will create models describing the intended deployment of their software, referring to the same set of stakeholder and basic asset types from the CSAP template. These models will be created using the System Security Modeller tool, and include the security properties of the software, taking account of its design, implementation and testing. The models will also incorporate any constraints that should be satisfied by the deployment, such as services that should be controlled
by the same stakeholder, or security measures that should be present in other assets interacting with the provided software.

4. Service operators will combine the overall template with models from their software providers to generate a design-time model of the envisaged system. At this stage we believe the CSAP tool should be used for this, since a template-based approach is more likely to ensure the service operator does not overlook important asset types or stakeholders (especially indirect stakeholders). However, the resulting model will then be analysed by the SSM tool, automatically finding threats and determining risk levels, taking account of countermeasures specified by the software providers.

5. Service operators will then add further security measures to the design-time model to bring risk levels from identified threats down to an acceptable residual risk level. There are two possible ways they could do this: by specifying security measures they expect to see, and then looking for cloud service providers whose security offer matches their requirements, or by choosing a cloud provider to provision the host for each service, importing security measures from that provider and checking whether the resulting deployment is satisfactory.

The idea is that the sub-models devised by software providers and cloud service providers should be created and analysed using SSM, allowing threats to be identified and the relationship between security measures and risk reduction from these threats can be made explicit. Penetration tests used by software providers should then ensure that the implementation is indeed able to resist those types of threats — i.e. the pen testers should emulate the threats found by analysing the sub-model. In this way we can ensure that security measures and penetration tests are related to threats that will be present in the overall system, and contribute directly to the service operator’s own risk assessment covered by the last two steps in the above procedure. Thus the relationship between ISO 27005 and ISO 15408 can be operated in reverse, by splitting the ISO 27005 part so it can be done partly in advance of penetration tests (by the software provider), and completed afterwards (by the service operator) using information from those tests.

Note that we are specifically interested in penetration tests because that is how one of the partners (OCC) currently verifies the security properties of their software. However, the above methodology generalises and can be used with other types of security verification, to any Evaluation Assurance Level (EAL) as defined by ISO 15408.

This procedure does involve a more complex interplay then we originally envisaged between CSAP (the RestAssured tool for using templates for guided asset-based risk analysis) and SSM (the RestAssured tool for automated risk identification and risk level evaluation). The process starts with a CSAP template which provides a common terminology, then technology providers use SSM to create machine understandable models of the components (including security properties) they offer. Then the service operator uses CSAP again to combine models w.r.t. the common template, and then SSM to perform the risk level evaluation and determine additional security controls required.

At this stage of the project, the RestAssured CSAP and SSM prototypes do not support such a sophisticated interplay. We have identified how the concepts should be mapped between the two views of the cloud-based system, and agreed how to exchange data. However, neither tool currently supports automatic refinement of a system component based on an imported sub-model, for example. This is a difficult problem to solve in a purely graphical environment, since any relationship to the initial (atomic) system component has to be distributed between sub-components in the refined model. Our experience is that only very simple cases are easily handled by users using graphical representations. We therefore plan to investigate whether these mappings could be inferred, and what extra information software providers or cloud service providers need to specify in their sub-models to enable this.
4.4 Design-Time versus Run-Time Risk Evaluation

Once the service operator is satisfied, they can deploy their services to the selected cloud providers. At that point, a run-time model will be created by the RestAssured adaptation component, which is updated to reflect any changes in configuration. The run-time model will be used to keep track of the relationships and security measures for assets in the running system, allowing risks to be evaluated on a continuous basis.

At this point, we intend that the RestAssured risk evaluator will alert the RestAssured adaptation system if it detects that a change in the system has caused risk levels to rise to the point where action is needed. The adaptation system may also trigger changes for other reasons. In either case, the RestAssured adaptation system will propose changes to the RestAssured risk evaluator, to check that risk levels will remain or return to acceptable levels before the change is implemented.

In RestAssured, risk assessment is model based (RestAssured deliverable D7.1) which facilitates the use of matching reasoning in risk assessment. Moreover, in RestAssured risk assessment is performed at both design-time, when a service is being designed, and at run-time while that service is operating. The run-time risk assessment combines the output of the design-time risk assessment performed in the System Security Modeller (SSM) tool (D7.1), with the run-time model of the system maintained by Adaptation (RestAssured deliverable D5.1).

The run-time risk model is the fusion of the design-time risk model and dynamic information from the monitored system. The information content of what the system ought to be, and how the components ought to be connected, is therefore completely defined within the design-time model. The relationship between the service operator and the technology suppliers is also defined in the design-time model, and information from technology suppliers (based on evaluating security properties) is used in the design-time risk analysis. Thus it should not be necessary to repeat security verification (specifically penetration tests) during the run-time phase. The interplay between ISO 27005 and ISO 15408 can be expressed and resolved at design-time, allowing the run-time procedure to focus on incremental updates according to ISO 27005 only.
5 Conclusion

This deliverable has described some foundational contributions to RestAssured:

- The architecture of the RestAssured run-time system, including its main components, the main context entities, and the main interfaces both among the RestAssured components and between RestAssured and its operational context.

- The testbed that provides the infrastructure for deploying, integrating, and testing the RestAssured components.

- The RestAssured methodology, combining design-time and run-time risk assessment approaches to achieve the high-level business objectives relating to compliance and continuous risk management.

These artifacts lay the groundwork for the further contributions of RestAssured, described in the deliverables of work packages 4 to 7. Each of these artefacts is the result of 16 months of joint efforts from the project partners, including several iterations of feedback, improvement, and consolidation.

In the second half of the project, further evolution of these artefacts is planned. In particular,

- The architecture is planned to be extended with aspects relating to multi-tenancy (multiple applications and multiple data controllers using the same RestAssured run-time system) and decentralization (eliminating the single point of failure and performance bottleneck of a centralized architecture).

- The methodology is planned to be further extended towards a seamlessly integrated process from service definition to service operation for all involved parties. This will also imply further extension to and integration of the existing CSAP and SSM tools.

Regarding the testbed, no significant changes are planned. However, the testbed should continue to be aligned with the deployment, integration, and testing needs of the project, so that changes to the used or supported technologies or to other technical circumstances may necessitate changes to the testbed as well.
Bibliography


