The research leading to these results has received funding from the European Community’s H2020 research and innovation programme under grant agreement n° 731678.
<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>Concept for Privacy Compliance Restoring Reconfigurations</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 June 2019</td>
</tr>
<tr>
<td>Actual Delivery Date</td>
<td>30 June 2019</td>
</tr>
<tr>
<td>Contributing WPs</td>
<td>WP5</td>
</tr>
<tr>
<td>Editor(s)</td>
<td>Zoltan Mann</td>
</tr>
</tbody>
</table>
# Contributors

<table>
<thead>
<tr>
<th>Name</th>
<th>Organization</th>
<th>Sections</th>
</tr>
</thead>
<tbody>
<tr>
<td>Julian Bellendorf</td>
<td>UDE</td>
<td>1, 4</td>
</tr>
<tr>
<td>Florian Kunz</td>
<td>UDE</td>
<td>2</td>
</tr>
<tr>
<td>Jan Laufer</td>
<td>UDE</td>
<td>3</td>
</tr>
<tr>
<td>Zoltan Mann</td>
<td>UDE</td>
<td>1–4</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>April 2019</td>
<td>Initial version</td>
</tr>
<tr>
<td>V0.2</td>
<td>June 2019</td>
<td>Content ready for cross-WP review</td>
</tr>
<tr>
<td>V0.3</td>
<td>June 2019</td>
<td>Improved version after cross-WP review</td>
</tr>
</tbody>
</table>
## Contents

**List of Figures** ........................................................................ 6  
**List of Tables** ......................................................................... 7  

1 **Introduction** ...................................................................... 8  

2 **Concept** ........................................................................... 9  
   2.1 Monitoring ......................................................................... 11  
   2.2 Analysis ............................................................................. 11  
   2.3 Planning ............................................................................. 11  
   2.4 Execution ........................................................................... 13  

3 **Implementation** .................................................................. 14  
   3.1 Changes since D5.1 .............................................................. 14  
   3.2 Overview of the technical implementation ........................... 14  
   3.3 Meta-Model ......................................................................... 14  
   3.4 Adaptation Logic ................................................................. 15  
   3.5 ServerApplication ............................................................... 16  
   3.6 Implementation status ......................................................... 16  

4 **Conclusion and future work** ................................................... 17
List of Figures

Figure 2.1: The Adaptation component and the Run-time model within the conceptual end-to-end architecture for data protection, as described in D3.3. ......................................................... 9

Figure 2.2: Overview of the activities performed in the Adaptation component, as a Data Flow Diagram. The activities are also mapped on the phases of the MAPE model. ......................................................... 10

Figure 2.3: Flowchart of the planning algorithm. ............................................................... 12

Figure 3.1: Implemented components and their relationships to other RestAssured components ...... 15
List of Tables

Table 3.1: Package overview of the implementation of the Adaptation component and the Run-time model ......................................................................................................................................................................................... 16
1 Introduction

This deliverable presents the results of Work Package WP5, “Run-time data protection assurance”, achieved between months 17 and 30 of the project. In terms of the overall RestAssured architecture defined in deliverable D3.3, this deliverable details the Adaptation component and the Run-time model.

The overall goal of WP5 is to deliver novel monitoring and adaptation solutions for detecting and mitigating data protection violations in the cloud. To this end, WP5 pursues the following objectives and is accordingly structured into the following tasks:

- Task 5.1: Detecting privacy policy violations
- Task 5.2: Restoring privacy policy compliance by means of adaptations
- Task 5.3: Models@runtime as a shared knowledge base

After we concentrated in D5.1 on the tasks T5.1 and T5.3, in accordance with the project plan laid out in the DoA, we focus in this deliverable on the task T5.2. However, progress could be achieved for all three tasks mentioned. The full approach of WP5 integrating the results of all three tasks will be presented in D5.3.

WP5 uses the concept of risk patterns (introduced in deliverable D5.1) for both detecting data protection violations and devising adaptations for mitigating data protection violations. In this context, there is a close interplay with run-time risk assessment developed in WP7 (described in deliverables D7.1 and D7.2). The run-time adaptation engine of WP5 considers also adaptations for goals other than data protection (e.g., to minimize costs); however, such adaptations are carried out only if the resulting configuration is deemed acceptable by the run-time risk assessment of WP7.

Adaptations elaborated in WP5 complement the work in WP4 and WP6 on data access protection according to data protection policies. While changes in individual data subjects’ data protection preferences are handled by policy changes (WP6) and by the effect of policy changes on data access (WP4), changes that affect the behavior of a whole system (e.g., because of changes in the environment) are covered by adaptations (WP5).

This deliverable is organized as follows. Chapter 2 describes, on a conceptual level, the approach created in WP5 for run-time data protection assurance by means of adaptations. Following the established MAPE (monitor – analyse – plan – execute) model of self-adaptive systems, we describe the activities performed and the artefacts created and used by the proposed approach, with special emphasis on the planning phase (which is the focus of Task 5.2).

Chapter 3 gives an update on the implementation of the Adaptation component and the Run-time model. The used technologies are described, as well as the refined package structure of our program.

Chapter 4 sums up the main points of the document and discusses conclusions as well as directions for future work.
2 Concept

Within the RestAssured run-time system, Adaptation is concerned with ensuring the continued satisfaction of data protection and other requirements despite changes. To this end, the managed systems (e.g., cloud infrastructure or applications) are monitored and a run-time model of the managed systems is updated so that it always mirrors the current state of the managed systems. The run-time model is searched for risks of data protection violation. If an intolerably high risk\(^1\) is identified, an adaptation is performed to mitigate the risk. This chapter describes the conceptual operation of the Adaptation component and the Run-time model, which are important elements of the RestAssured run-time system (see Figure 2.1).

Searching for risks is done on two levels. On the first level, the Adaptation component itself performs a quick search in the Run-time model using the concept of risk patterns introduced in D5.1. On the second level, the Run-time risk assessment component of WP7 performs a more in-depth analysis using the techniques described in D7.1 and D7.2. Through this separation, the Adaptation component by itself can figure out an adaptation that will likely lead to a good configuration in terms of data protection risks. The adaptation proposal has to be approved by the Run-time risk assessment component before the adaptation is actually executed. The two-level analysis of risks allows us to reduce the communication between the Adaptation component and the Run-time risk assessment component to the necessary minimum, thus significantly improving the performance of the overall RestAssured run-time system.

There are two main objectives for the Adaptation component. The primary objective is to mitigate high risks. The secondary objective is to improve the satisfaction of other functional and non-functional requirements (e.g., increasing efficiency, decreasing costs) of the managed systems. Since there can be multiple different adaptation possibilities in a given situation, we aim for choosing the “best” adaptation. Several criteria can be used to evaluate the different adaptation possibilities and to find the best one:

- The set of functionalities available in the target configuration

---

\(^1\)In this document, the term “risk” always refers to a risk of data protection violation
- Non-functional properties of the target configuration (e.g., financial costs and performance)
- The impact of the change from the current to the target configuration (e.g., amount of transient overhead or transient performance drop)

The concept of the Adaptation component follows the general structure of the MAPE model. There are four phases in this model: monitor, analyse, plan and execute. Our structure differs slightly from the general MAPE model because the analyse and plan phases overlap in our approach (as shown in Figure 2.2).

Figure 2.2: Overview of the activities performed in the Adaptation component, as a Data Flow Diagram. The activities are also mapped on the phases of the MAPE model.

In the monitoring phase, the Adaptation component listens for changes in the managed systems. The configuration of every managed system is stored in the Run-time model. If a change is detected, the model is updated and the analysis phase starts by checking the model for risks. Planning is invoked if a risk has been found. To eliminate the risks, the Adaptation component generates several possible adaptations, which
are immediately checked for risks by the Adaptation component. Those adaptations that – based on the quick internal risk checking – mitigate the risk, are evaluated in terms of their impact on other goals. Starting with the best, the generated adaptation proposals are sent to the Run-time risk assessment component, until there is one that is approved by Run-time risk assessment. With the best adaptation found, the execution phase starts. The chosen adaptation is enacted on the managed system.

In the following, the four phases in the activities of the Adaptation component are described in more detail.

2.1 Monitoring

The managed systems (including cloud infrastructure, cloud applications, or RestAssured components) provide monitoring information about their configuration through a uniform interface, described in D5.1.

Every managed system registered with the Adaptation component has an adapter which provides an interface, that allows to request information considering the managed system’s current configuration. The monitoring information received is converted into a model, based on the meta-model for run-time models introduced in D5.1, and stored in the Run-time model. The old model of the managed system is then compared to the new model. If changes are detected, the new model has to be analysed.

2.2 Analysis

If the run-time model is changed, it has to be checked for risks. The first search for risks is provided internally by the Adaptation component, using the concept of risk patterns introduced in D5.1. A risk pattern models a partial configuration of the managed systems that exhibits a potential data protection violation. Manifestations of these risk patterns are searched for in the run-time model through the use of a pattern matching algorithm. If a risk is found, the planning phase is invoked to generate adaptation proposals. If no risk is found by the Adaptation component, the model is sent to the Run-time risk assessment component, which carries out a more in-depth analysis. If the Run-time risk assessment component finds a risk in the new model, it informs the Adaptation component via a specified interface, which also triggers the planning phase within the Adaptation component.

2.3 Planning

Planning is invoked when the current configuration of the managed system incurs an intolerably high data protection risk. The aim of planning is to devise an adaptation that mitigates the risk. The adaptations proposed are either atomic adaptations or compound adaptations. A compound adaptation consists of a number of atomic adaptations. Atomic adaptations are the smallest adaptations generated by the Adaptation component. Since some risks cannot be mitigated through the use of a single atomic adaptation, multiple atomic adaptations can be combined to a compound adaptation, which may be more effective in mitigating the risks.

There are three kinds of atomic adaptations:

- Creating or removing an entity
- Changing the value of an attribute of an entity
- Changing a relation between entities

These atomic adaptations can be applied to some types of entities, attributes and relations, but not to all types. For example, an entity of the type “Virtual Machine” can be created as an adaptation; however,
an entity of the type “Physical Machine” cannot be created. Thus a list with the allowed types of atomic adaptations is introduced. This list currently holds a default set of entities, attributes and relations, but will later be customizable through a specified interface.

In general, there can be several adaptations that would mitigate the found risks. These adaptations have to be evaluated to find the best one. For this purpose, criteria like performance and cost are inspected. Furthermore it is checked, if the currently available functionality is still available after the adaptation is executed. To enable the comparison of configurations with different sets of available functionalities, it is possible to define how important the different functions are, by giving each function a value of importance. The higher the value, the more important the function is. Adaptations that preserve more important functions are thus rated higher.

![Flowchart of the planning algorithm](image)

**Figure 2.3: Flowchart of the planning algorithm**

At the start of the planning phase (see Figure 2.3) a list containing every atomic adaptation that can be applied in the current configuration is created. These atomic adaptations are then grouped into possible compound adaptations. There are several possible methods for creating the compound adaptations, for example:

- Generating all possible combinations of atomic adaptations
- Generating all possible combinations of at most $k$ atomic adaptations, where $k$ is a given threshold
- Combining atomic adaptations that belong to a set of known sensible combinations
- Considering combinations that were successful in the past
For each considered adaptation, the configuration that would result from the adaptation is checked for
risks by means of the risk patterns. The result of this step is a list with the possible adaptations that mitigate
the risks found previously and do not lead to significant new risks. The adaptations on the list are then
evaluated and sorted from best to worst with respect to the specified criteria (functionality and non-functional
goals like costs). For the purpose of sorting, a scoring function can be used that combines the considered
quality attributes to a single metric.

Starting with the best, the adaptation proposals are sent to the Run-time risk assessment component one
by one. The reply of the Run-time risk assessment component is a “green light” if the proposed adaptation
indeed leads to a good configuration in terms of data protection risks. Since the adaptation proposals are
processed from the top of the list, the first adaptation accepted by Run-time risk assessment is the best found
adaptation. Thus, if the reply is a green light, the execution phase is initiated and the remaining possible
adaptations are not sent for in-depth risk assessment. However, if the Run-time risk assessment component
rejects the current adaptation proposal, the planning algorithm continues with the next possible adaptation
on the list.

Remark. A challenge for the planning algorithm is that the number of potential adaptations can be very
large. In this case, processing all possible adaptations would take a very long time. To get this problem
under control, a possible solution is to limit the number of adaptations to process. In this way an effective
adaptation (which may not be the best though) can be found in acceptable time. This is important since
the discovered risks could cause great damage, so it is important to mitigate them quickly. Afterwards, the
search for further adaptations, which would bring the system to an improved configuration, can be continued.

2.4 Execution

If an adaptation was decided, it is executed on the managed system through the appropriate interface (defined
in D5.1). Once the adaptation is executed on the managed system, the monitoring recognises the changes
made and updates the Run-time model.
3 Implementation

This chapter outlines the main characteristics of the current status of the implementation of the Adaptation component and the Run-time model within the overall RestAssured architecture. The focus is on how the meta-model is implemented, how the run-time model is stored, and the technologies used to realize the adaptation algorithm. Moreover, the changes in the package structure that have been performed since D5.1 are also described.

3.1 Changes since D5.1

Since Deliverable 5.1, we have restructured the application. On the one hand, we changed the technology for storing persistent data. This also influences the way the meta-model is created. On the other hand, the program structure has changed slightly due to reorganization. These changes were carried out to simplify the implementation of the adaptation logic. The packages are now different in structure and names, but still offer the same functions. The definition of the interfaces described in D5.1 has not changed, only the logic behind the interfaces has been extended and optimized.

Previously, the run-time model was implemented using object-relational database mapping (ORM). The Hibernate ORM framework was a simple way to map the meta-model of the run-time model to a relational database schema and store the run-time model. Instead of Hibernate, we now use the Eclipse Modelling Framework (EMF) to create the meta-model at design time and use it at run time. EMF enjoys wide-spread adoption in the model-based software engineering community. The advantages of EMF include an easy-to-use meta-model editor that can generate Java code. Furthermore, EMF is supplemented by a rich ecosystem of related tools. One of these tools is Henshin which we use in the implementation of the adaptation logic. Henshin supports graph pattern matching and model transformation based on an EMF model.

3.2 Overview of the technical implementation

As shown in Figure 3.1, the software created in WP5 is separated into three components. The functionality of these components is summarized in Table 3.1. The individual components are described in the next three sections.

3.3 Meta-Model

We designed the meta-model with the EMF (Eclipse Modelling Framework). The EMF is a modelling framework and tool for creating Java source code based on a structured data model. Furthermore, the EMF combines three important technologies: UML, Java, and XML. EMF generates the source code which forms the basis for the run-time model. The run-time model contains service instances (i.e., models of the current configuration of managed systems) and service adaptation proposals (i.e., proposed new configurations for the managed systems). Service instances and service adaptation proposals are based on the same meta-model. We also use EMF to store service instances and service adaptation proposals persistently as files in XMI format. XMI (XML Metadata Interchange) is a standard for exchanging meta-data information via Extensible Markup Language (XML). Service instance and service adaptation proposal can be described as follows:

- ServiceInstance: A ServiceInstance represents a running instance of the managed system (application or infrastructure service) at run time.
• ServiceAdaptationProposal: A ServiceAdaptationProposal represents an adaptation of the managed system that has been proposed, but has not necessarily been executed. A ServiceAdaptationProposal does not represent a specific change but it describes a target configuration. It is based on the same meta-model as a ServiceInstance. ServiceAdaptationProposals are needed to propose possible adaptations to the Run-time risk assessment component developed in WP7.

Moreover, we implemented an additional transformation functionality to convert from EMF to JSON format, so that we can provide service instances and service adaptation proposals in JSON format to other RestAssured components, thus continuing to serve the existing interfaces. The UML, XMI and JSON support allows a general use of our software.

As both EMF and the meta-model described in D5.1 are compliant with UML, it was possible to implement the meta-model from D5.1 in EMF with just a few adjustments. EMF represents a graph as a tree structure, hence a root node had to be added to the meta-model. This root node is called “CloudEnvironment”.

To represent relations in XMI and JSON, EMF uses IDs that consist of meta-model-specific class names and run-time-generated object identifiers. These IDs are unique for each object of a service instance. We are using the IDs, for example, to identify objects that are marked from risk assessment as a part of a threat.

### 3.4 Adaptation Logic

The adaptation logic uses the Henshin model transformation library for finding and realizing possible adaptations. “Henshin provides a state-of-the-art model transformation language for the Eclipse Modeling Framework.”\(^1\) Henshin uses transformation rules that are made of two parts: Left-hand side and right-hand side. The left-hand side of a transformation rule captures the pattern that should be found in a model. The right-hand side of the transformation rule specifies how parts of the model that match the pattern should

---

\(^1\)https://www.eclipse.org/henshin/
Table 3.1: Package overview of the implementation of the Adaptation component and the Run-time model

<table>
<thead>
<tr>
<th>Package name</th>
<th>Functionality</th>
<th>Relation to external components</th>
</tr>
</thead>
<tbody>
<tr>
<td>MetaModel</td>
<td>• Providing meta-model (generated by EMF)</td>
<td></td>
</tr>
<tr>
<td>AdaptationLogic</td>
<td>• Creating service adaptation proposals</td>
<td></td>
</tr>
<tr>
<td>ServerApplication</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ServerApplication.restController</td>
<td>• Communication with Risk Assessment</td>
<td>• Risk Assessment</td>
</tr>
<tr>
<td></td>
<td>• Providing monitoring interface</td>
<td>• Managed systems</td>
</tr>
<tr>
<td></td>
<td>• Providing execution interface</td>
<td></td>
</tr>
<tr>
<td>ServerApplication.runtime</td>
<td>• Creating and storing run-time service instance models</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Storing service adaptation proposals</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Invoking adaptation logic</td>
<td></td>
</tr>
</tbody>
</table>

be transformed. It is possible to capture all relevant cloud adaptations as Henshin rules. These rules are used to create service adaptation proposals that should be examined by risk assessment. Every adaptation proposal is saved as a temporary object and as a file in XMI format. If the Run-time risk assessment component greenlights an adaptation proposal, we update the relevant service instance and execute the changes on the managed system.

3.5 ServerApplication

The ServerApplication has two main roles: it is responsible for providing the Run-time model (storing service instances and service adaptation proposals) and for providing the REST interfaces to other components.

After a managed system (application or infrastructure service) has registered with the RestAssured run-time system, a service instance is created for the managed system. This service instance represents the run-time model of the managed system. A service instance is based on the EMF meta-model. The models of individual managed systems are identified by unique IDs within the run-time model. In addition, a service instance is also saved as a file in XMI format to ensure the availability of the data for example for diagnosis purposes.

Service adaptation proposals are stored in the same way. Every service adaptation proposal is related to a managed system. Moreover, every service adaptation proposal gets a unique identifier.

The RestController within the ServerApplication is responsible for providing the REST interfaces to other RestAssured components (in particular, Run-time risk assessment) and other systems (in particular, the managed systems). The interfaces are described in D5.1.

3.6 Implementation status

The status of the implementation can be summarized as follows: The adaptation logic is in place and can create several ServiceAdaptationProposals. These proposals are sent one after the other to Run-time risk assessment until one of them is greenlighted. Moreover, monitoring and execution infrastructure is in place to manage a useful set of systems.

We are currently implementing the prioritization of adaptations in terms of costs and functionality as described in Chapter 2. Besides we are implementing adapters for monitoring and adaptation of specific managed systems for validation and demonstration purposes.
4 Conclusion and future work

In this deliverable, we presented the concept for run-time data protection assurance by means of adaptations. The most important features of our approach can be summarized as follows:

- Model-based: a run-time model of the managed systems is updated by means of monitoring. The run-time model is analyzed to identify high risks of data protection violations. Also possible adaptations are evaluated based on the model.

- Compound adaptations: beyond single atomic adaptations, also compound adaptations (consisting of multiple atomic adaptations) are considered. Compound adaptations are potentially much more powerful than single atomic adaptations and can hence mitigate a larger set of data protection risks.

- Prioritization of adaptations: from the set of possible adaptations that can mitigate a given data protection risk, the approach selects the best one, taking into account both functional and non-functional requirements.

- Approval by Run-time risk assessment: an adaptation is executed only if Run-time risk assessment has approved it.

We also presented the status of the implementation. In particular, we showed the transition to a new set of technologies based on the Eclipse Modeling Framework and the Henshin model transformation library. These technologies allow a more direct and more effective implementation of the adaptation logic. However, the interfaces to other components have remained largely stable.

The future work in WP5 aims at finalizing the implementation of all steps of the proposed approach, ensuring an integrated operation. Moreover, the validation of the approach is also a focus for the remaining work.