



Methods and tools for GDPR Compliance through  
**P**rivacy and **D**ata **P**rotection **4E**ngineering

Specification and design of requirements  
engineering tool for privacy and data protection  
**V1**

Project: PDP4E  
Project Number: 787034  
Deliverable: D4.1  
Title: Specification and design of requirements  
engineering tool for privacy and data protection  
Version: v1.0  
Date: 16/07/2019  
Confidentiality: Public  
Author: Patrick Tessier  
Gabriel Pedroza  
Nicolás E. Diaz Ferreyra

Funded by



# Table of Contents

<b>DOCUMENT HISTORY .....</b>	<b>4</b>
<b>LIST OF FIGURES.....</b>	<b>5</b>
<b>ABBREVIATIONS AND DEFINITIONS.....</b>	<b>5</b>
<b>EXECUTIVE SUMMARY .....</b>	<b>6</b>
<b>1 INTRODUCTION .....</b>	<b>7</b>
<b>1.1 OBJECTIVE OF THE DOCUMENT .....</b>	<b>7</b>
<b>1.2 STRUCTURE OF THE DOCUMENT.....</b>	<b>7</b>
<b>1.3 RELATION WITH OTHER DELIVERABLES .....</b>	<b>7</b>
<b>2 BACKGROUND ON REQUIREMENT ENGINEERING TOOL.....</b>	<b>9</b>
<b>2.1 ECLIPSE.....</b>	<b>9</b>
<b>2.2 PAPHYRUS.....</b>	<b>9</b>
2.2.1.1 Editors.....	10
2.2.1.2 Model explorer .....	10
2.2.1.3 Property View .....	10
2.2.1.4 Possibility to support DSL and plug new tools .....	10
<b>2.3 PAPHYRUS-REQ.....</b>	<b>10</b>
2.3.1 Features of Papyrus-Req for Requirements Management .....	11
2.3.1.1 Requirements specification .....	11
2.3.1.2 Requirement Research .....	11
2.3.1.3 Validation.....	11
2.3.1.4 Statistics.....	11
2.3.1.5 Import/export.....	11
2.3.2 Visualization.....	12
2.3.2.1 Model explorer .....	12
2.3.2.2 Editor .....	12
2.3.3 Papyrus-Req extended for Privacy and Data Protection.....	12
<b>2.4 PROPAN .....</b>	<b>13</b>
<b>3 USER NEEDS.....</b>	<b>15</b>
<b>4 REQUIREMENTS ELICITATION .....</b>	<b>16</b>
<b>5 USE CASES .....</b>	<b>21</b>
<b>6 DESIGN .....</b>	<b>22</b>
<b>6.1 PDP4E REQUIREMENT ARCHITECTURE .....</b>	<b>22</b>
6.1.1 ProPan Taxonomies Module .....	23

6.1.1.1	Description.....	23
6.1.1.2	Input .....	23
6.1.1.3	Output .....	23
6.1.1.4	Requirement satisfaction .....	23
6.1.2	GDPR Taxonomies Module .....	23
6.1.2.1	Description.....	23
6.1.2.2	Input .....	24
6.1.2.3	Output .....	24
6.1.2.4	Requirement satisfaction .....	24
6.1.3	Requirement-Designer Module.....	24
6.1.3.1	Description.....	24
6.1.3.2	Input .....	24
6.1.3.3	Output .....	24
6.1.3.4	Requirement satisfaction .....	25
6.1.4	Validation Module .....	25
6.1.4.1	Description.....	25
6.1.4.2	Input .....	25
6.1.4.3	Output .....	25
6.1.4.4	Requirement satisfaction .....	25
6.1.5	Privacy-MetaReq-DB Module .....	25
6.1.5.1	Description.....	25
6.1.5.2	Input .....	26
6.1.5.3	Output .....	26
6.1.5.4	Requirement satisfaction .....	26
6.1.6	GDPR-MetaReq-DB Module .....	26
6.1.6.1	Description.....	26
6.1.6.2	Input .....	26
6.1.6.3	Output .....	26
6.1.6.4	Requirement satisfaction .....	26
6.1.7	Privacy-Analysis Module.....	27
6.1.7.1	Description.....	27
6.1.7.2	Input .....	27
6.1.7.3	Output .....	27
6.1.7.4	Requirement satisfaction .....	27
6.1.8	GDPR-Analysis Module .....	27
6.1.8.1	Description.....	27
6.1.8.2	Input .....	27
6.1.8.3	Output .....	27
6.1.8.4	Requirement satisfaction .....	28
<b>7</b>	<b>SUMMARY.....</b>	<b>29</b>
<b>8</b>	<b>REFERENCES .....</b>	<b>30</b>

## Document History

Version	Status	Date
V0.1	Draft	11/06/2019
V0.2	Contribution of Nicolas	17/06/2019
V0.4	Contributions from Gabriel Pedroza. Reviewing and modifications.	26/06/2019
V0.5	Modify version numbers. Add content to respond to the review	28/06/2019
V0.7	Review by Victor Muntés (Beawre)	12/07/2019
V0.8	Review by Estíbaliz Arzoz Fernández (Trialog)	12/07/2019
V1.0	Finalization	16/07/2019

Approval		
	Name	Date
Prepared	Patrick Tessier	28/06/2019
Reviewed	Victor Muntés (Beawre), Estíbaliz Arzoz Fernández (Trialog)	12/07/2019
Authorised	Antonio Kung	17/07/2019
Circulation		
Recipient	Date of submission	
Project partners	28/06/2019	
European Commission	17/07/2019	

## List of Figures

Figure 1: Links with other work-packages .....	8
Figure 2: Papyrus overview .....	9
Figure 3: UML4PF tool realization overview [11] .....	13
Figure 4: Generation and validation of privacy requirements using the ProPan tool [11].....	14
Figure 6: Functional requirements endowed to the PDP4E-Req tool .....	16
Figure 7: Use Cases .....	21
Figure 8: Architecture Overview.....	22

## Abbreviations and Definitions

Abbreviation or Terms	Definition
ICT	Information and Communication Technologies
DSL	Domain Specific Language
EPL	Eclipse Public License
fUML	Executable UML
OCL	Object Constraint Language
MARTE	Modelling and Analysis of Real-time and Embedded systems
Papyrus	UML modeller inside Eclipse
Papyrus-Req	Customization of Papyrus to manage Requirements
PDP4E	Privacy and Data Protection 4 Engineering
PDP4E-Req	Tool resulted from WP4 to management GDPR and privacy requirement
ProPan	Problem-based Privacy Analysis
PSCS	Precise Semantics of UML Composite Structures
ReqIF	Requirements Interchange Format
SysML	Systems Modeling Language
UML	Unified Modeling Language
UDEPF	University Duisburg-Essen Problem Frames

## Executive Summary

This document describes the architecture of the PDP4E framework for elicitation of requirements oriented to privacy and data protection in systems and software projects. The tool architecture supports the method specified in D4.4 [1] which takes into account the legal obligations introduced by the EU General Data Protection Regulation (GDPR). Since the approach is inspired in the Problem-based Privacy Analysis (ProPan) [2] method<sup>1</sup>, the architecture inherits a subset of its features and implemented modules. The overall architecture is named PDP4E-Req and integrates a subset of requirement taxonomies as defined in ProPan, and also introduces new taxonomies specific to GDPR.

Overall, the core contributions of this deliverable are:

- A –yet to evolve– list of user-oriented requirements guiding the specification and development of PDP4E-Req
- A first draft of functional requirements to guide the implementation of the PDP4E-Req architecture
- An overview and description of the overall PDP4E-Req architecture and modules

---

<sup>1</sup> ProPan was originally developed by researchers at the University of Duisburg Essen (UDE)

# 1 Introduction

## 1.1 Objective of the document

The purpose of this document is to present the architecture of PDPE-Req. This tool is a support to the methodology defined as a report in D4.4[1]. This tool allows on the one hand to specify the conventional requirements: functional and non-functional described in most requirements-based approaches. On the other hand, this tool will bring added value to specify requirements around privacy, requirements directly from the GDPR. This specification will be based on modeling techniques to specify the necessary information for the description of privacy and GDPR domains as well as the execution of analysis in order to generate a corpus of requirements necessary for the design of compliant systems.

The architecture presented in this document is based upon UML and SysML modeling. The Use-Case and Composite Diagram are also used.

The tool is based on existing technologies: Eclipse platform<sup>2</sup>, Papyrus<sup>3</sup>, Papyrus-Req and ProPAN [2]. These technical bases will be briefly explained in the document.

## 1.2 Structure of the document

The document is structured into the following sections. Section 2 focuses on the technological bases on which PDP4E-Req is developed. We will quickly present Eclipse and Papyrus, then in more details Papyrus-Req and Propan. These descriptions provide a structural overview of those technologies. Section 3 will list the functional requirements after this analysis to support the methodology presented in Deliverable D4.4. These requirements are mostly inherited from that deliverable. Section 4 presents the use cases and the grosgrain architecture of the tool. The detailed list of requirements specifying the architecture design comes in Section 6. A summary of this specification is given in Section 7.

## 1.3 Relation with other deliverables

WP4 focuses on specification of requirements. It is related to other work-package and therefore other deliverables as it is shown in *Figure 1*.

---

<sup>2</sup> <https://www.eclipse.org>

<sup>3</sup> <https://www.eclipse.org/papyrus/>

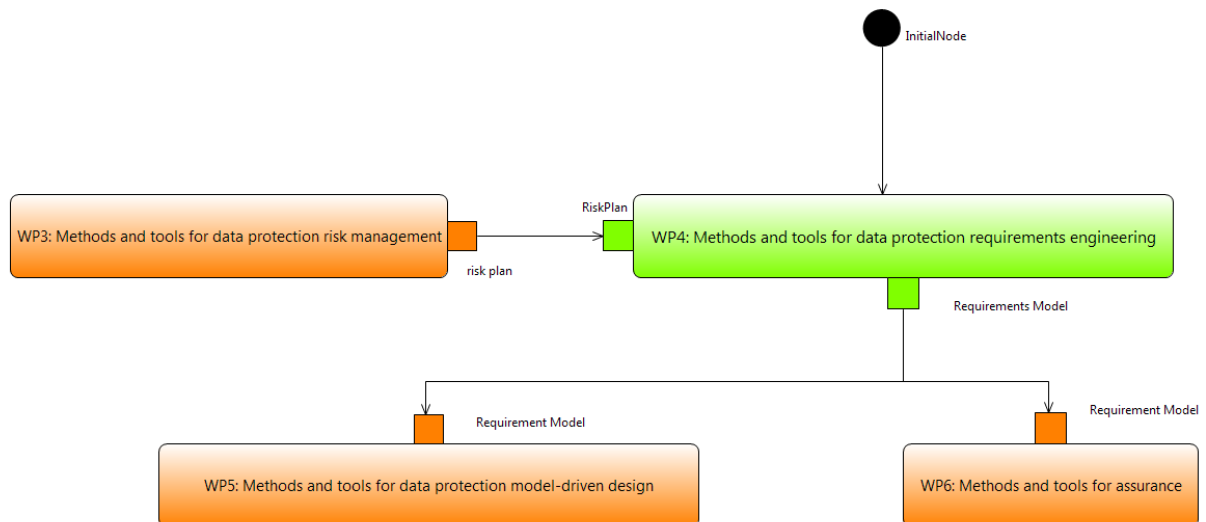


Figure 1: Links with other work-packages

The architecture specified in this deliverable is primarily related to the tools developed in WP3: *Method and Tools for data protection risk management*. An analogous link has been also established in previous work with safety or security approaches [3], [4]. The outcomes of the tools developed in WP3 will be an entry of PDP4E-Req. These risk mitigations could become new non-functional requirements or new technical requirements to implement. This risk mitigation import will be one of the aspects to be addressed in upcoming deliverables.

The WP4 tool architecture is also related to WP5 tools: *Methods and tools for data protection model-driven design*. The link with WP5 tools makes sense in an engineering approach. After having the specification of the requirements, the detailed specification is carried out [5], [6]. PDP4E-Req (tool resulted from WP4 to management GDPR and privacy requirement) will provide a list of requirements that will be used as input for WP5 tools. In addition, PDP4E-Req will provide traceability support tools required for model refinement activities.

The PDP4E tool for requirements has a relationship with the tools of WP6: *Methods and tools for assurance*. On the one hand, in order to realize assurance, PDP4E-Req will be able to provide the intra- requirements traceability links and links with its design and implementation (WP5). On the other hand, PDP4E-Req will provide the traceability links with the modeled GDPR to ensure that the requirements truly correspond to the legal text.

Finally, the architecture specified in this report is guided by the requirements elicited from stakeholder needs and is summarized in D2.4, Overall system requirements. The system requirements consider the current state of practice (methods and tool support) to achieve privacy and data protection.



## 2 Background on requirement Engineering tool

### 2.1 Eclipse

Eclipse is a foundation created to benefit the providers of software development offerings and end-users by allowing a vendor neutral, open, and transparent community to be established around the Eclipse projects. More details can be found in the following report [7].

Eclipse provides also a technical platform. See the following explanation from [8].

*“It defines the set of frameworks and common services that collectively make up infrastructure required to support the use of Eclipse as a component model, as a Rich Client Platform (RCP) and as a comprehensive tool integration platform. These services and frameworks include a standard workbench user interface model and portable native widget toolkit, a project model for managing resources, automatic resource delta management for incremental compilers and builders, language-independent debug infrastructure, and infrastructure for distributed multi-user versioned resource management.”*

The eclipse Platform is our technical basis in order to develop PDP4E-Req tool. The main reason for this is that Papyrus-Req is developed on top of Eclipse and we have selected it as a background tool to be leveraged for the purposes of PDP4E.

### 2.2 Papyrus

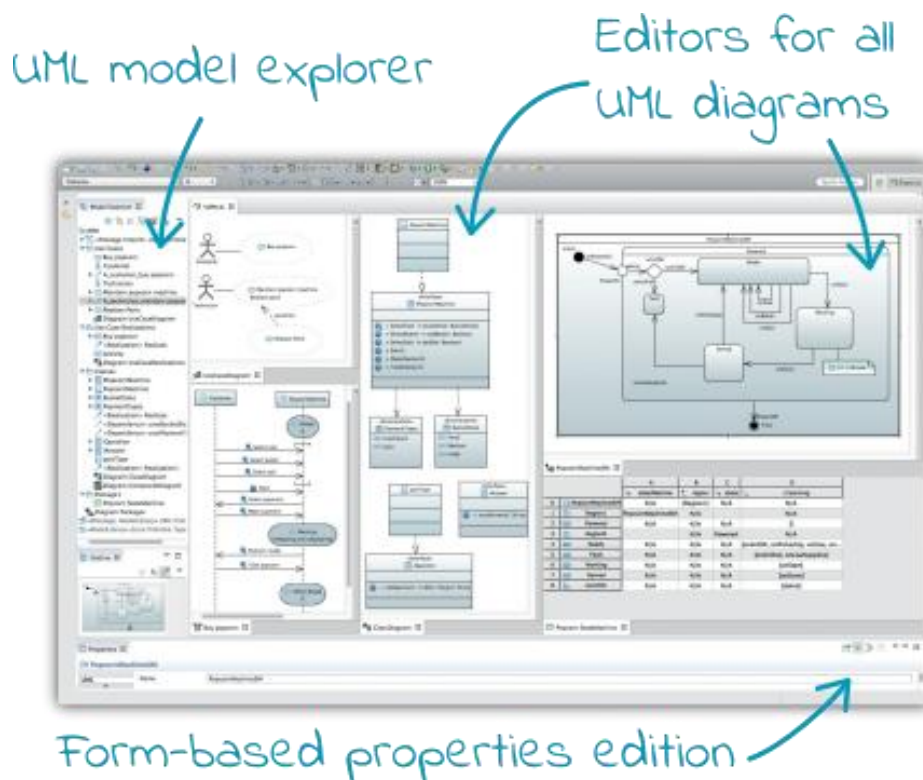


Figure 2: Papyrus overview

Papyrus is a modeling environment implementing the standard modeling languages UML, SysML, MARTE or OCL. Papyrus has been built on top of the Eclipse Platform. It offers an EPL (Eclipse Public License) implementation of the OMG standards precisely specifying the operational semantics of these languages as the fUML standard setting the semantics of the models based

on the class and activity diagrams, as well as its extensions defining the semantics of PSCS component models or the forthcoming standard defining the semantics of state machines. Thus, Papyrus allows the execution of models and more generally offers an environment allowing the automatic exploitation of the models (simulation, analysis and synthesis) for engineering purposes for the systems and software design.

### **2.2.1.1 Editors**

Papyrus is a tool consisting of several editors, mainly graphical editors but also completed with other editors such as textual-based and tree-based editors. All these editors allow simultaneous viewing of multiple diagrams of a given UML model. Modifying an element in one of the diagrams is immediately reflected in other diagrams showing this element. Papyrus is integrated in Eclipse as a single editor linked to one UML2 model. Model editors can be arranged in tabbed views, and several tabbed views can be arranged side by side (left, right, top and bottom). Such views can be created, re-sized, moved, and deleted by dragging them to the desired position. Papyrus editors are highly customizable.

### **2.2.1.2 Model explorer**

Papyrus has got a model explorer that displays elements contained in the model, editors that reference elements.

The order and the representation of contained element is dynamic and can be changed by using a configuration file. Thanks to this, it is possible to see an element with different representation for several domains.

This model explorer can also display several menus to interact with Papyrus and diagrams.

The menu to create elements are declarative and can be updated according to a specific domain.

### **2.2.1.3 Property View**

The property view contains attributes to edit the model elements but also to edit their graphical appearance.

The structure of each property is defined in a declarative way and can be adapted for each specific domain.

### **2.2.1.4 Possibility to support DSL and plug new tools**

As explained before, the Papyrus editors can be customized for each specific domain. Papyrus has got internal mechanisms to manage the life cycle of modelling elements. That is mandatory to support the definition of Domain Specific Languages (DSL). For example, it is possible to precise the behaviour of an element at creation, during movement and deletion actions.

Papyrus also supports CSS to customize appearance of all diagrams.

## **2.3 Papyrus-Req**

The Papyrus-Req tool is a Papyrus add-on that completes the implementation of the SysML standard within Papyrus. SysML is a dedicated standard for systems modeling. The standard makes possible to specify the requirements and the links necessary links for traceability. Nevertheless, these elements are specified in a too generic way and are not fully adapted for the tasks in the requirements management process in the native Papyrus.

The Papyrus-Req module was developed to facilitate the management of SysML requirements in the Papyrus environment. Thus, the Papyrus-Req is mainly committed to improve requirements management and support the display of requirements.

### **2.3.1 Features of Papyrus-Req for Requirements Management**

Papyrus-Req aims to improve requirements management by better supporting the creation of requirements, their validation, their search, the ability to display statistics, and to import or export information from files in standardized formats.

#### **2.3.1.1 Requirements specification**

Papyrus-Req improves the user experience for requirements specification by providing dedicated menus and a set of keyboard shortcuts. These mechanisms for requirements creation are based upon a creation policy module. The creation policy makes it possible to define a unique identifier for each requirement, by adding a prefix number and, if necessary, a default text.

The policy to create requirements is also configurable based on user preferences. For instance, it is possible to configure the prefix, set the number of digits and finally specify the separator between the container requirements and the sub-requirements.

#### **2.3.1.2 Requirement Research**

The requirement model can be quite large and can contain a large number of requirements [9]. In order to help the user, it is possible to search among requirements by their identifier or relying on words contained in the requirements description. This search can be conducted in the explored model or within all Papyrus dialogs.

#### **2.3.1.3 Validation**

Papyrus-Req offers specialization mechanisms for validation of requirements according to the following:

- Check that a requirement is met by an architectural element (also called *satisfy* mechanism)
- Check that a requirement is validated by an element representing a test (also called *verify* mechanism).

Once the mechanisms are applied for a given requirement, its validation is displayed to the user by a dedicated icon that appears in the element within the explorer model and in the Papyrus editors.

#### **2.3.1.4 Statistics**

Papyrus-Req also allows to display in a dedicated view the statistics in the form of histogram or pie chart that displays the requirements rate satisfied or verified.

Other evaluation axes can be added for statistics according to user needs.

#### **2.3.1.5 Import/export**

Finally, Papyrus-Req allows to import-export requirements relying upon the ReqIF [10] or CSV formats.

The ReqIF import function in Papyrus-Req not only imports the requirement letting it “as is”. It conducts a transformation of the imported element considering a UML profile that includes types and categories of requirements as specified in the ReqIF format.

The CSV format allows Papyrus-Req to interface with tools like MS-Excel, which is broadly used by the requirements community.

## 2.3.2 Visualization

Papyrus-Req aims to enhance the visualization of requirements mainly targeting model usability. To do so, the improvement of model exploration and navigation are addressed via different user interfaces and editors.

### 2.3.2.1 Model explorer

Basically, the way requirements are displayed in Papyrus can hide relevant details of a requirement. Indeed, the graphical representation of requirements is based generically only on the names of the elements. The identifiers and the description are both important elements for the users. Indeed, engineers may need to have a quick understanding of the requirement's contents. Papyrus-Req addresses this problem by customizing the explorer model to display the identifier and the associated description. This makes it possible to have a tree view of the requirements at a glance through the explorer model. Another important feature brought by Papyrus-Req is the display of traceability. Papyrus allows displaying in this tree the downward or upward traceability. Downward traceability makes it possible to identify a requirement within all the refinement sub-trees or even architectural elements that indirectly satisfy the selected requirement. In a similar way, the upward traceability makes it possible to display the sub-tree of the refined requirements. If this view is applied in an architecture model, it is possible to visualize all the requirements satisfied by each architecture element in the model.

### 2.3.2.2 Editor

Papyrus-Req implements editors to visualize the requirements in tabular form: a requirement can be displayed as flat row of a table or as a tree. In addition, Papyrus-Req provides a set of matrix editors to display the traceability links. A validation link matrix is also available to structure requirements verification fulfilment and refinements via check boxes. Thanks to the Papyrus basis upon which Papyrus-Req relies, it is possible to add other specialized editors.

## 2.3.3 Papyrus-Req extended for Privacy and Data Protection

Papyrus-Req is a tool that will be used as a technological basis to cover WP4 objectives.

In particular, it will be extended to support the edition of requirements oriented to address the specificities identified in GDPR for privacy and data protection.

The traceability proposed and already supported by Papyrus-Req seems essential to address these aspects and, more specifically, the need for tool-support to perform refinements that are traceable, consistent, easy-to-navigate and manageable by engineers. These features are to be ensured not only within the framework for requirements management (WP4) but also within the framework for privacy and data protection by design (WP5).

As previously mentioned, Papyrus-Req provides functions for importing and exporting requirements. This module will be important to import, for instance, subparts of risks produced from the tools developed in WP3. Preserving traceability in exported models is an important feature, as long as exported requirements can be an input of the tools developed in WP5.

Finally, Papyrus-Req is a technological basis that will ease the definition and implementation of new model validation rules. This feature is useful, for instance, to validate well-formed models.

## 2.4 ProPan

ProPan is a model-driven requirement engineering method whose purpose is the generation of *privacy requirements* from the *functional requirements* of software systems. That is, it analyses systematically the functionality of a system-to-be in order to identify privacy related issues and generate the corresponding requirements in a computer-aided fashion. In ProPan, a system-to-be is expected to be modelled following the Problem Frame approach introduced by Jackson [13]. Such approach allows the representation of contextual information, interfaces and phenomena related to the system's operational environment. For this reason, ProPan is implemented as an extension of UML4PF<sup>4</sup>, a UML-based tool which supports the generation of problem diagrams as defined by Jackson.

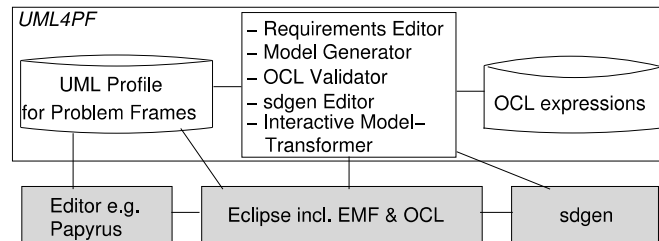


Figure 3: UML4PF tool realization overview [11]

At its core, UML4PF consists of a profile that represents the Jackson's original notation through UML elements. This profile is then integrated in an Eclipse plugin that facilitates the development and analysis of problem diagrams. For this purpose, validation conditions in OCL are incorporated in the UML4PF plugin that performs semantic checks on the developed diagrams. In this way, UML4PF supports software engineers in developing a coherent and complete set of requirements documents. The overall architecture of UML4PF is illustrated in Figure 3 and consists of:

- A UML Profile for Problem Frames that defines relevant stereotypes such as <<ProblemDiagram>>.
- A Requirements Editor which allows to add new requirements.
- A Model Generator to automatically generate model elements like observed and controlled interfaces.
- An OCL Validator that checks if the model is valid and consistent by evaluating OCL expressions. Likewise, it also returns the location of invalid parts of the model.
- The *sdgen* editor used to edit sequence diagrams.
- An Interactive Model Transformer for creating software architectures through interactive model transformations.

ProPan's tool support is built over the foundations of UML4PF and developed with the Eclipse Modelling Framework (EMF) and the Sirius Framework<sup>5</sup>. Such tool support is called UDEPF (University Duisburg-Essen Problem Frames) and consists of an EMF meta-model and a graphical editor to create, modify, and provide views on instances of the meta-model. The ProPan tool that assists the application of the ProPan method uses instances of the UDEPF meta-model as input. Particularly, it incorporates taxonomies of privacy requirements which help to identify privacy-relevant information flows in a system's model. Moreover, these taxonomies that are derived from the GDPR and the ISO 29100 support the generation of privacy requirement instances for the system under analysis. Figure 4 illustrates the ProPan tool and some of its core functions such as the generation of data-flow diagrams, generation of privacy requirements and

<sup>4</sup> <http://www.uml4pf.org>

<sup>5</sup> <https://www.eclipse.org/sirius/>

identification of personal data. Both, UML4PF and ProPAN, have been applied on a case study developed by the industrial partners of the EU project *Network of Excellence on Engineering Secure Future Internet Software Services and Systems (NESSoS)* [12]. This scenario is based on the German healthcare system which uses health insurance schemes for the accounting of treatments.

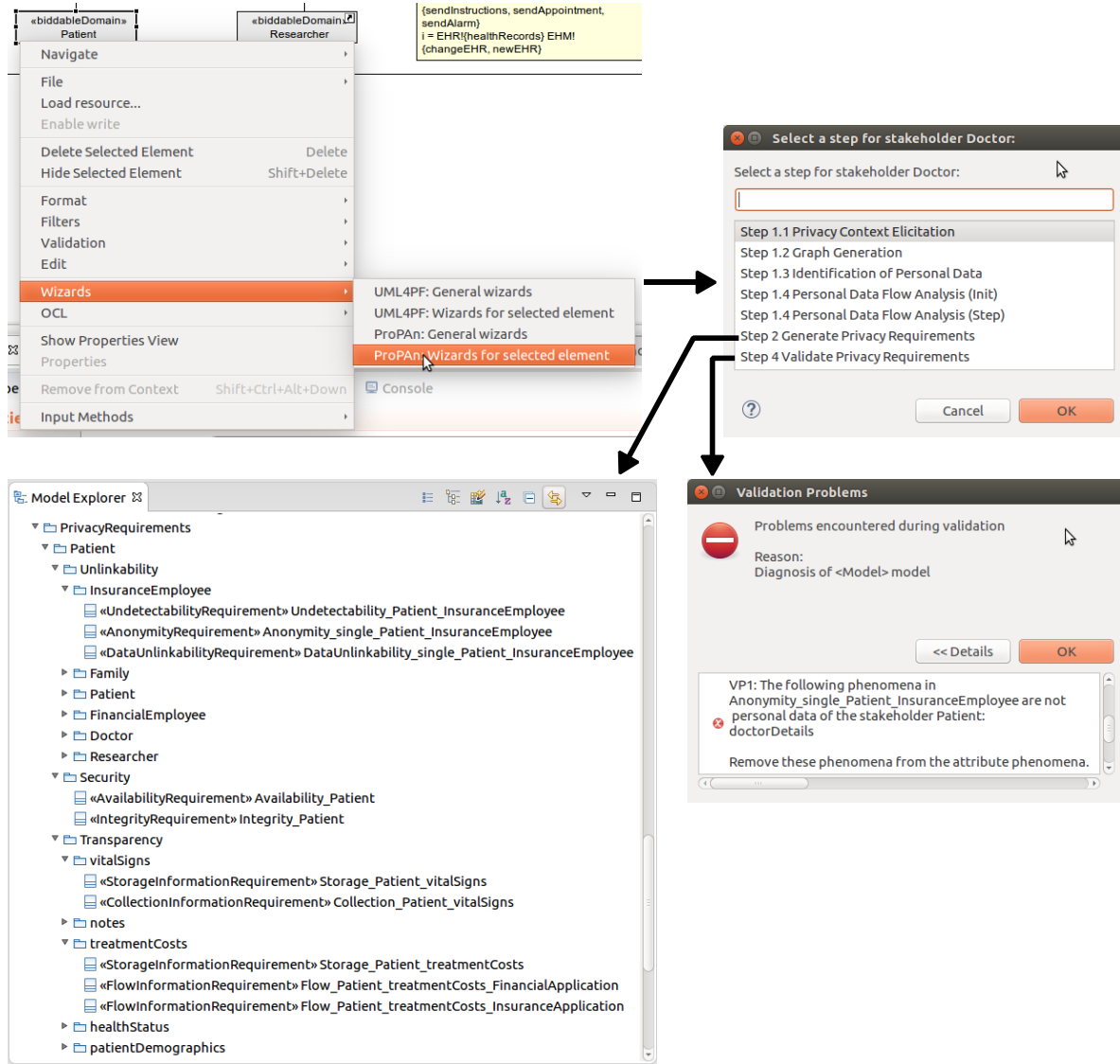


Figure 4: Generation and validation of privacy requirements using the ProPAN tool [11]

### 3 User needs

Requirements related to privacy and data protection must be satisfied by any system or service involved in data processing activities and in particular when personal or sensitive data are involved. But first, it needs to be analysed the **scope of the requirements**, depending on the nature, scope, context, purpose and risks of the data processing activities (Art. 24.1, Art. 35.1). That is, it needs to be ascertained **which requirements** need to be abided by a systems or software development project. First analyses conducted on GDPR point out that some of the following aspects shall be considered during referred ascertainment:

- the categories of personal data i.e. how sensitive data is (Rec. 51, Art. 9),
- the size of the organization with regards to e.g. record keeping (Rec. 13, Art. 30),
- the application of corporate policies viz. binding corporate rules (Art. 47),
- the application of specific processing types such as profiling (Art. 22),
- the purposes for which the personal data was provided (Rec. 50, Art. 5.1.b, Art. 5.1.e),
- whether the data subject may have objected to some purposes e.g. direct marketing (Art. 21.2, Art. 21.3),
- special purposes such as research (Rec. 53, Rec. 156, Rec. 159, Rec. 162, Art. 89),
- the lawfulness and legitimacy of the bases for processing (Rec. 47, Art. 6), etc.

Plus, the requirements need to be **instantiated** regarding the specific scope and features of the system-to-be (e.g. the concept of “data transfers” shall be considered in particular for each and every of the data transfer operations that are planned for a specific system).

Tools exist that provide support to the **compliance operationalization**, that is, the definition of the detailed requirements that are applicable in the context of an organization to comply with the law, including the possibility to introduce **customized rule-sets** derived from other regulatory sources (such as corporate rules, codes of conduct, etc. as above mentioned). The architecture specified in this deliverable shall evolve according the evolution of users/stakeholders needs and related requirements as specified in D2.4, *Overall systems requirements*.

## 4 Requirements elicitation

The specification of the tool follows a model driven design approach. Thus, in this section, the functional requirements for the design will be first explained. Since we are currently at the initial phase of the specification, the list of requirements appears to be short. Of course, this list will be completed and refined all along the implementation phase.









«Requirement»  AddFunctionalReq id=PDP4E-Req01 text=PDP4E-Req shall provide for the Requirement Engineer the ability to add / edit functional requirements	«Requirement»  AddNonFunctionalReq id=PDP4E-Req02 text=PDP4E-Req shall provide for the Requirement Engineer the ability to add / edit non-functional requirements
«Requirement»  ImportReq id=PDP4E-Req03 text=PDP4E-Req shall provide for the Requirement Engineer the ability to import risk a requirement goals	«Requirement»  AddSpecificReq id=PDP4E-Req04 text=PDP4E-Req shall provide for the Requirement Engineer the ability to add / edit specific information dedicated to the GDPR
«Requirement»  GenerateReq id=PDP4E-Req05 text=PDP4E-Req shall provide for the Requirement Engineer the ability to generate requirement dedicated to the privacy and GDPR domain	«Requirement»  AddTraceability id=PDP4E-Req06 text=PDP4E-Req shall provide for the Requirement Engineer the ability to add / edit traceability between elements in the model
«Requirement»  UpdateMetaReq id=PDP4E-Req07 text=PDP4E-Req shall provide for the developer customize the ability to update meta-Requirements	«Requirement»  validateModel id=PDP4E-Req08 text=PDP4E-Req shall provide for the Requirement Engineer the ability to validate the model.

Figure 5: Functional requirements endowed to the PDP4E-Req tool

<b>PDP4E-Req01</b>	<b>AddFunctionalReq</b>
	<b>WP4</b>
Description	PDP4E-Req shall provide for the Requirement Engineer the ability to add / edit functional requirements
Assigned WP	WP4
Relation to other requirements	



Actor	Requirement Engineer
Priority	Must Have
Type	Functional
Non-functional category	
Rationale	The foundation of the tool covering basic functional need

<b>PDP4E-Req02</b>	<b>AddNonFunctionalReq</b>
	<b>WP4</b>
Description	PDP4E-Req shall provide for the Requirement Engineer the ability to add / edit non-functional requirements
Assigned WP	WP4
Relation to other requirements	
Actor	Requirement Engineer
Priority	Must Have
Type	Functional
Non-functional category	
Rationale	A non-functional requirement is also a requirement so it must be possible to add a non-functional requirement

<b>PDP4E-Req03</b>	<b>ImportReq</b>
	<b>WP4</b>
Description	PDP4E-Req shall provide for the Requirement Engineer the ability to import subparts of risk mitigations as requirement goals
Assigned WP	WP4
Relation to other requirements	PDP4E-Req02
Actor	Requirement Engineer
Priority	Should have

Type	Functional
Non-functional category	
Rationale	The risk analysis for privacy and data protection can produce goals, refined as requirements which can guide the design of the system

<b>PDP4E-Req04</b>	<b>AddSpecificReq</b>
	<b>WP4</b>
Description	PDP4E-Req shall provide for the Requirement Engineer the ability to add / edit specific information related to the GDPR regulation
Assigned WP	WP4
Relation to other requirements	PDP4E-Req01
Actor	Requirement Engineer
Priority	Should have
Type	Functional
Non-functional category	
Rationale	In order to achieve privacy and data protection as demanded by GDPR, specific elements pertaining to requirements should be added / edited

<b>PDP4E-Req05</b>	<b>GenerateReq</b>
	<b>WP4</b>
Description	PDP4E-Req shall provide for the Requirement Engineer the ability to generate requirements dedicated to the privacy and GDPR domain
Assigned WP	
Relation to other requirements	PDP4E-Req01
Actor	Requirement Engineer
Priority	Must have
Type	Functional
Non-functional category	

Rationale	This is the most important functionality added by this module. It covers the generation (assisted or semi-automatic) of requirements to achieve privacy and data protection as demanded by GDPR.
-----------	--

<b>PDP4E-Req06</b>	<b>AddTraceability</b>
	<b>WP4</b>
Description	PDP4E-Req shall provide for the Requirement Engineer the ability to add / edit traceability between elements in the model
Assigned WP	WP5
Relation to other requirements	PDP4E-Req01
Actor	Requirement Engineer
Priority	Must have
Type	Functional
Non-functional category	
Rationale	This functionality supports the designer by ensuring that requirements are implemented and are traceable

<b>PDP4E-Req07</b>	<b>UpdateMetaReq</b>
	<b>WP4</b>
Description	PDP4E-Req shall provide for the developer customizer the ability to update meta-requirements
Assigned WP	
Relation to other requirements	PDP4E-Req05
Actor	Requirement Engineer
Priority	could have
Type	Functional
Non-functional category	
Rationale	If the GDPR is modified, the metaReq should be updated accordingly to the modidications.

<b>PDP4E-Req08</b>	<b>ValidateModel</b>
	<b>WP4</b>
Description	PDP4E-Req shall provide for the Requirement Engineer the ability to validate the model.
Assigned WP	
Relation to other requirements	PDP4E-Req05
Actor	Requirement Engineer
Priority	could have
Type	Functional
Non-functional category	
Rationale	In order to be sure that generated requirements are correct, the model written by the engineer must be correct (e.g., well-formed, well-defined, well-structured)

## 5 Use cases

In the UML use case diagram in Figure 6, relationships between PDP4E-Req actors/stakeholders and use cases are illustrated. Use cases are deduced from the functional requirements described previously in Section 4.

In PDP4E, the partners in this work-package aim to develop not only a “once-and-for-all” tool but rather a platform that can be customized according to the specific needs of the application domain, stakeholders needs, etc. For instance, a customization can be made considering context and needs within a systems development company.

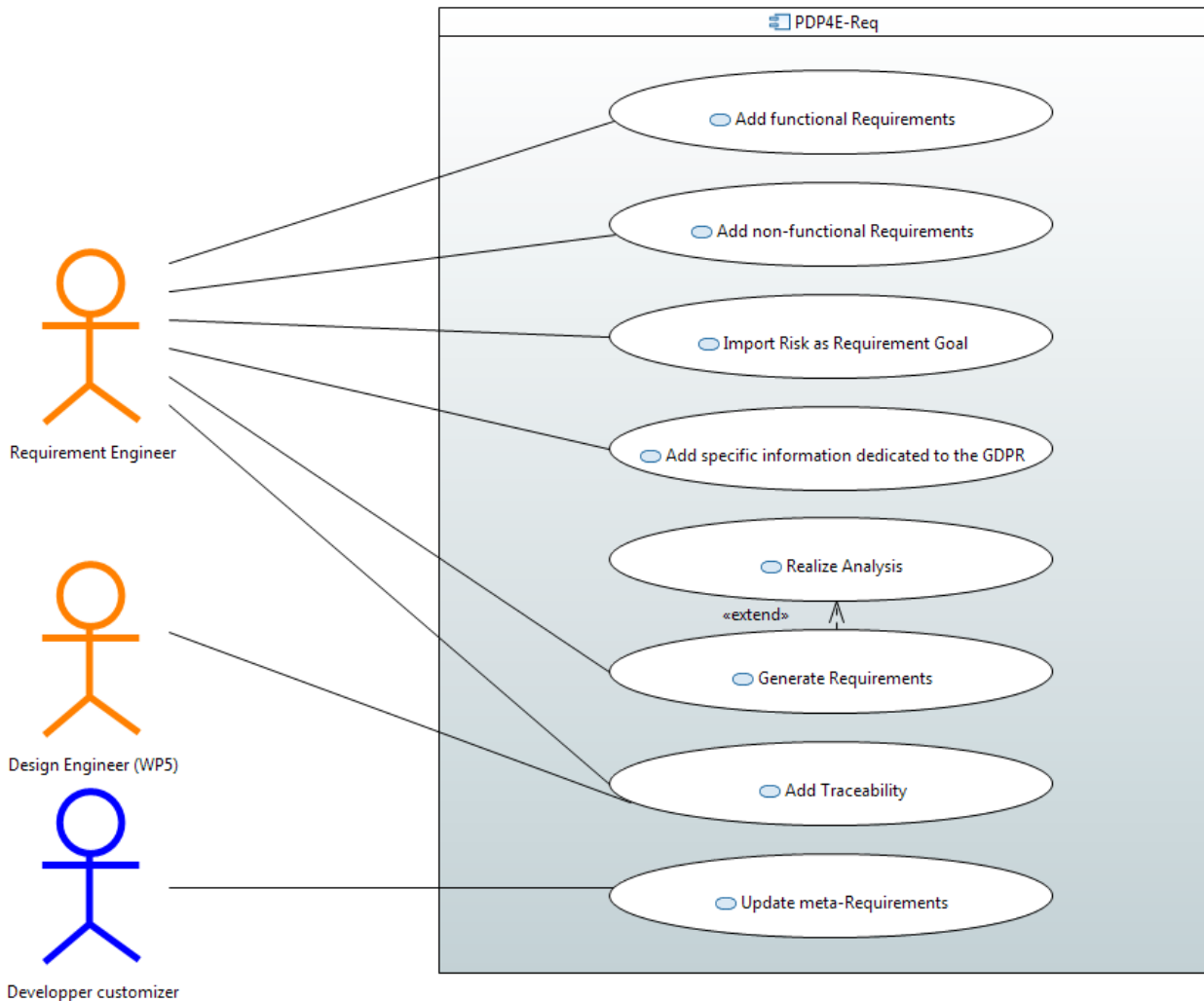


Figure 6: Use Cases

As shown in Figure 6, 3 types of actors exist:

- The requirements engineer who can create functional and non-functional requirements. Requirement Engineer can also import risk mitigations, specify non-functional requirements related to privacy and data protection as in GDPR. He can also generate requirements and add traceability elements.
- Notice that the design engineer, foreseen in WP5, will also be able to use the traceability tools between the requirements and the detailed design elements of the system.
- The developer who customizes PDP4E-Req for a specific company context, application domain, etc. in addition to the person in charge of maintaining the requirement database according to the evolutions of the GDPR.

## 6 Design

### 6.1 PDP4E Requirement Architecture

As previously mentioned, the PDP4E-Req architecture is an extension of Papyrus-Req that integrates and reuses the most salient features of the ProPan tool.

PDP4E-Req architecture incorporates ProPan's architectural modules and respective principles. The definition of the PDP4E-Req architecture is as follows. First, ProPan already deploys modules to integrate a set of taxonomies, and to specify the information about data context, data flows, and privacy-related issues of a system-to-be. These modules come with diagram editors to ease model's edition. In addition, ProPan includes a module to infer from context and system related diagrams the candidate requirements to achieve privacy, relying upon taxonomies. In PDP4E, we will develop a set of modules that extend the taxonomies defined in ProPan relying upon meta-requirements that cover the specificities from GDPR. The referred modules should generate requirements related to privacy and data protection in conformity with GDPR.

As it is specified in D4.4, the meta-requirements are modeling structures amenable to be instantiated as concrete requirements. More specifically, the meta-requirements include ready-to-be-filled elements (delimited by placeholders) that can be completed for a specific context and system design instance.

Nevertheless, it is recalled that PDP4E-Req architecture does not leverage all ProPan modules. More specifically, PDP4E-Req will not include all ProPan diagrams in order to simplify the inferring process (thus saving time) and to render the approach more friendly-usable to engineers. To do so, a single Data Flow Diagram will be specified conveying relevant system-to-be and context information. Last but not least, PDP4E-Req will provide modules to complement the existing ProPan analysis to cover and ensure alignment with chapter 2 and 3 of the GDPR.

The diagram in Figure 7 presents, in a macroscopic way, the intended architecture of PDP4E-Req. This global view is represented using a UML composite diagram. The model view allows to represent the links between the inputs and outputs of each module. This coarse-grained view does not show details about the internal structure of each module. A certain heterogeneity is nonetheless observed since there are modules that can be decomposed into several Eclipse plug-ins whereas others are indeed singleton plug-ins.

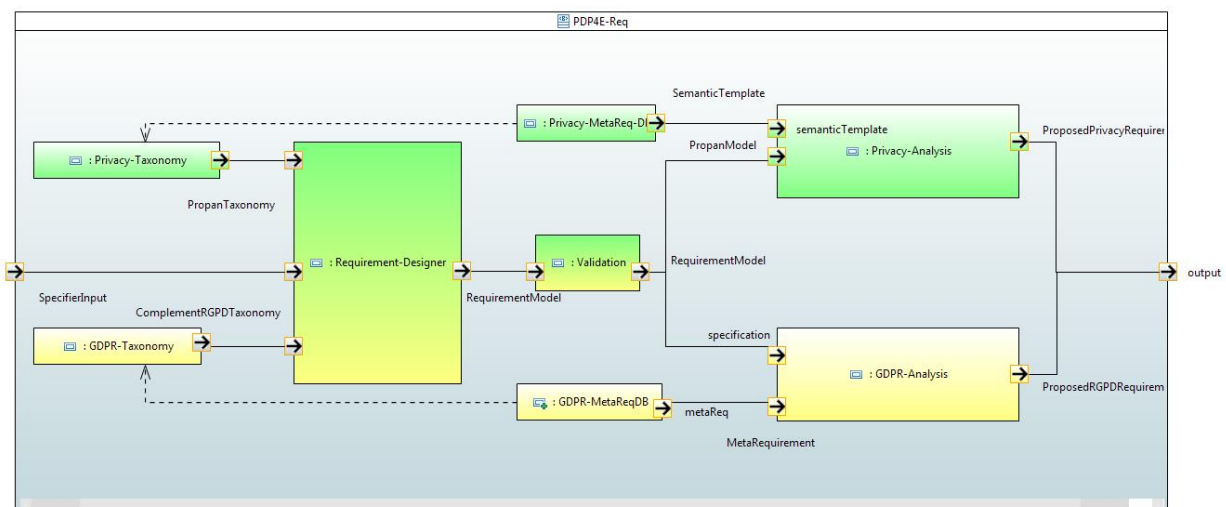


Figure 7: Architecture Overview

In this composite diagram, two symmetrical-colored parts can be distinguished: the yellow and green parts. All green modules are mostly inherited from the ProPan tool (with some simplifications and adaptations). All yellow modules represent additional modules dedicated to address specific aspects of GDPR (to be developed in PDP4E). Finally, the modules which are both yellow and green represent the front-end editors of PDP4E-Req. The referred modules are two-colored not just for explanation purposes: the front-end editor must indeed integrate the 2 analysis.

The above architecture distribution obeys to the following criteria. On the one hand, we aim to preserve the analysis of concerns as they are defined by ProPan whereas its most-salient features selected for PDP4E are still maintained. On the other hand, we search for more flexibility at front-end but also transparency during the analysis of concerns. Whereas the separation of modules seems adequate at architectural level, the complementarity of concerns analyses is still to be ensured. Indeed, we still need to elaborate on that respect, for instance, via a case study in which both native ProPan and specific PDP4E-Req features are aligned to elicit privacy and data protection requirements. For those cases, the proposed architecture results flexible enough because it reduces the interdependencies between modules. Low dependency between module functions shall allow, for instance, parallel work between engineer teams which can distribute tasks either on native ProPan or on PDP4E-Req.

Detailed descriptions of PDP4E-Req modules are given in the following subsection.

## 6.1.1 ProPan Taxonomies Module

### 6.1.1.1 Description

This module consists of needed plug-ins to specify/model the taxonomies in ProPan. The taxonomies inherit and implement privacy principles as they were introduced by Hansen. Once implemented, the elements supported by this module can be used directly by users through editors or called from backend to generate privacy requirements.

The taxonomies shall be implemented in the context of UML. To do so, the taxonomies will be specified relying upon a UML profile: a profile specializes the semantics of UML. This taxonomy module will also include a set of libraries. Since they are independent, the libraries can be replaced or modified without the need for changing the profile.

### 6.1.1.2 Input

- No input

### 6.1.1.3 Output

- Technical API to manipulate the ProPan taxonomies.

### 6.1.1.4 Requirement satisfaction

- PDP4E-Req07: PDP4E-Req shall provide for the developer or customizer the ability to update the meta-requirements

## 6.1.2 GDPR Taxonomies Module

### 6.1.2.1 Description

This module consists of a set of Eclipse/Papyrus plug-ins necessary to support the definition of new taxonomies, complementary to ProPan, and to address specificities in chapters 2 and 3 of

the GDPR. The elements in the taxonomies can be directly used by users/engineers through PDP4E-Req editors. The new taxonomies can be called from backend to generate GDPR-related requirements.

This module is implemented in the context of UML. To do so, this taxonomies module will be implemented as a UML profile: a profile specializes the semantics of a UML language. This module will also include model libraries. The libraries can be replaced or modified without the need of changing profile.

#### **6.1.2.2 Input**

- No Input

#### **6.1.2.3 Output**

- Technical API to manipulate the GDPR specific taxonomies.

#### **6.1.2.4 Requirement satisfaction**

- PDP4E-Req07: PDP4E-Req shall provide for the developer customizer the ability to update meta-requirements

### **6.1.3 Requirement-Designer Module**

#### **6.1.3.1 Description**

The Requirement Designer module is very important since it essentially contains the user interface.

This module supports the graphical elements implemented by the backend modules which are interfaced with the modeling editors. The module will support in particular a Data Flow editor useful to specify processes and data exchanged within and between them. The Data Flows will be specified at high-level which is necessary to address the complexity of systems and certain privacy analyses. The Data Flows shall support annotations needed to store the information related to GDPR specificities, the system context, and the analyses outcomes, when adequate. The requirement designer module will be developed on top of Papyrus-Req editors which allow the edition of functional and non-functional requirements. Referred editors will be specialized so as to allow the edition of the different categories of requirements: *transparency, lawful, fair*. The categories will be defined in terms of the properties of principles as defined in GDPR. The module shall also allow the edition of such properties.

#### **6.1.3.2 Input**

- Profiles specified in the taxonomies module
- Interactions performed by the user to specify the system.

#### **6.1.3.3 Output**

- Model containing functional and non-functional requirements within a high-level data flow diagram specifying the processes and data involved.



#### **6.1.3.4 Requirement satisfaction**

- PDP4E-Req05: PDP4E-Req shall provide for the Requirement Engineer the ability to generate requirements dedicated to the privacy and GDPR domain
- PDP4E-Req07: PDP4E-Req shall provide for the developer customizer the ability to update the meta-requirements

### **6.1.4 Validation Module**

#### **6.1.4.1 Description**

The requirement specification tasks are sensitive since mistakes during specification can lead to many costly errors during implementation. In order to detect potential errors as early as possible, the idea is to check if the specified model is well-formed and in compliance with the syntax and structure inherited from taxonomies. This validation may help the engineer to complete the model when missing parts are identified or even to correct it. The validation module will interactively provide warnings or detected errors. The validation can be performed at any time during the requirements design.

#### **6.1.4.2 Input**

- Requirements model designed by the user

#### **6.1.4.3 Output**

- Validated model including detected warnings and errors

#### **6.1.4.4 Requirement satisfaction**

- PDP4E-Req05: PDP4E-Req shall provide for the Requirement Engineer the ability to generate requirement dedicated to the privacy and GDPR domain
- PDP4E-Req08: PDP4E-Req shall provide for the Requirement Engineer the ability to validate the model.

### **6.1.5 Privacy-MetaReq-DB Module**

#### **6.1.5.1 Description**

This module is a technological knowledge base used to generate requirements as defined in ProPan: it contains a set of meta-requirements. The meta-requirements are model structures containing ready-to-be-filled parts (fields or texts) that will be instantiated by the engineer according to a user-specified model of the system-to-be. This module is developed independently of the analysis modules for several reasons:

- As long as the analysis techniques can evolve, a need for updating the meta-requirements appear.
- The meta-requirements can also be adapted, or new ones added according to the context of the company and to take into account its specificities.
- The evolutions in meta-requirements and analysis techniques can be conducted independently and harmonized afterwards if needed.

This module will contain meta-requirements oriented to the privacy and data protection domain as specified in ProPan.

#### **6.1.5.2 Input**

- No input

#### **6.1.5.3 Output**

- List of meta-requirements oriented to privacy and data protection as defined in ProPan

#### **6.1.5.4 Requirement satisfaction**

- PDP4E-Req05: PDP4E-Req shall provide for the Requirement Engineer the ability to generate requirement dedicated to privacy and data protection as specified in ProPan
- PDP4E-Req07: PDP4E-Req shall provide for the developer customizer the ability to update meta-requirements

### **6.1.6 GDPR-MetaReq-DB Module**

This module is similar to the previous but dedicated to cover requirements from GDPR.

#### **6.1.6.1 Description**

This module is a technological knowledge base used to generate requirements as derived from GDPR: it contains a set of meta-requirements. The meta-requirements are model structures containing ready-to-be-filled parts (fields or texts) that will be instantiated by the engineer according to a user-specified model of the system-to-be. This module is developed independently of the analysis module for several reasons:

- As long as the analysis techniques can evolve, a need for updating the meta-requirements appear.
- The meta-requirements can also be adapted, or new ones added according to the context of the company and to take into account its specificities.
- The evolutions in meta-requirements and analysis techniques can be conducted independently and harmonized afterwards if needed.

This module will contain meta-requirements oriented to the privacy and data protection domain as specified in GDPR.

#### **6.1.6.2 Input**

- No input

#### **6.1.6.3 Output**

- List of meta-requirements oriented to cover GDPR specificities.

#### **6.1.6.4 Requirement satisfaction**

- PDP4E-Req05: PDP4E-Req shall provide for the Requirement Engineer the ability to generate requirement dedicated to privacy and data protection as specified in GDPR
- PDP4E-Req07: PDP4E-Req shall provide for the developer customizer the ability to update meta-requirements

## 6.1.7 Privacy-Analysis Module

### 6.1.7.1 Description

This module is used to generate the requirements from the privacy meta-requirements as defined in ProPan and the user-specified model of the system-to-be.

It will consist of analyzing and generating the privacy requirements according to the patterns applied to the user model.

### 6.1.7.2 Input

- Base of meta-requirements defined in ProPan
- Model designed by the engineer of the system-to-be

### 6.1.7.3 Output

- Set of privacy-oriented requirements

### 6.1.7.4 Requirement satisfaction

- PDP4E-Req05: PDP4E-Req shall provide for the Requirement Engineer the ability to generate requirements oriented to privacy relying upon meta-requirements in ProPan and a user-defined model of the system-to-be
- PDP4E-Req08: PDP4E-Req shall provide for the Requirement Engineer the ability to validate the user-defined model

## 6.1.8 GDPR-Analysis Module

### 6.1.8.1 Description

This module is analogous to the previous module but dedicated to cover specificities of GDPR. The GDPR-Analysis module is intended to generate requirements related to the GDPR and according to the meta-requirements instantiated on a user-defined model of the system-to-be.

It will perform an analysis in order to generate specific GDPR requirements. On one side, the generated requirements will refer to the articles of GDPR, on the other side, they will be linked to elements within the user-defined model referred in the requirements body.

From this generation, the requirements will be refined to a level adequate for design and implementation phases. The traceability between generated and refined requirements will be ensured. In this way, an engineer can search and identify the requirements generated and the articles at the origin of each requirement.

This approach will be essential in order to demonstrate the compliance with the GDPR to the authorities.

### 6.1.8.2 Input

- Base of meta requirement elicited from GDPR
- The model of the system-to-be developed by the engineer

### 6.1.8.3 Output

- Set of generated GDPR requirements specific to the user-defined model

#### **6.1.8.4 Requirement satisfaction**

- PDP4E-Req05: PDP4E-Req shall provide for the Requirement Engineer the ability to generate requirements oriented to privacy and data protection as defined in GDPR
- PDP4E-Req08: PDP4E-Req shall provide for the Requirement Engineer the ability to validate user-defined the model.

## 7 Summary

This document contains a first specification of the PDP4E architecture for the management of requirements oriented to privacy and data protection. The architecture is named PDP4E-Req and mainly integrates and harmonizes (1) salient features and modules selected from ProPan and (2) new modules developed on top of Papyrus-Req to cover the specificities found in GDPR. The approach and architecture are originally inspired in ProPan and introduce new modules to define taxonomies of requirements specific to GDPR. Both, the ProPan and GDPR taxonomies are the basis upon which requirements are elicited taking a model of the system-to-be as design target. The specification structures several items: stakeholders' needs, functional requirements fulfilling user-needs, and specific modules to comply with the functional requirements. Along with the harmonization between existing ProPan and extensions of Papyrus-Req, the PDP4E-Req architecture leverages Model Driven Engineering mechanisms in order to ensure essential features of the requirements management phase, like traceability, consistency, refinement, and validation. Since requirements management is a primary phase in the development cycle, the referred features are necessary to ensure exchanges and interoperability with other tools developed in PDP4E (design, risks, assurance). This specification will evolve according to the implementation and consolidation phases of PDP4E.

## 8 References

- [1] N. D. Ferreyra, P. Tessier, and G. Pedroza, "Requirements engineering methods for privacy and data protection v1," Deliverable D4.4.
- [2] K. Beckers, S. Faßbender, M. Heisel, and R. Meis, "A Problem-Based Approach for Computer-Aided Privacy Threat Identification," in *Privacy Technologies and Policy*, 2014, pp. 1–16.
- [3] S. Myagmar, A. J. Lee, and W. Yurcik, "Threat modeling as a basis for security requirements," in *Proceedings of the IEEE Symposium on Requirements Engineering for Information Security*, 2005.
- [4] D. Firesmith, "Specifying reusable security requirements.," *J. Object Technol.*, vol. 3, no. 1, pp. 61–75, 2004.
- [5] Michael Ryan, University of New South Wales, Canberra, Australia, Lou Wheatcraft, Requirements Experts, USA, Rick Zinni, Harris Corporation, USA, Jeremy Dick, Integrate Systems Engineering Ltd, UK, and Kathy Baksa, Pratt & Whitney, USA, *Guide for Writing Requirements 2.1*. San Diego, California 92111-2222 USA: INCOSE, 2017.
- [6] "System Requirements - SEBoK." [Online]. Available: [https://www.sebokwiki.org/wiki/System\\_Requirements](https://www.sebokwiki.org/wiki/System_Requirements). [Accessed: 31-Jan-2019].
- [7] P. Krief, "Open-source licensing model and IPR governance framework," Deliverable D7.5, Oct. 2018.
- [8] L. Vogel, J. Bresson, and D. Roy, "Eclipse Platform project," 24-Mar-2016. [Online]. Available: <https://wiki.eclipse.org/Platform>.
- [9] "JFIE 2018 – CFTL – Comité Français des Tests Logiciels." [Online]. Available: <http://www.cftl.fr/jfie-2018/>. [Accessed: 10-Jun-2019].
- [10] "Requirements Interchange Format (ReqIF) version 1.2," Object Management Group (OMG), formal/2016-07-01, 2016.
- [11] I. Côté, M. Heisel, H. Schmidt, and D. Hatebur, "UML4PF - A tool for problem-oriented requirements analysis" in *IEEE 19th International Requirements Engineering Conference*, 2011, pp. 349-350.
- [12] Network of Excellence (NoE) on Engineering Secure Future Internet Software Services and Systems (NESSoS), [Online]. Available: <http://http://www.nessos-project.eu/>. [Accessed 3 June 2019]
- [13] M. Jackson, *Problem frames: analysing and structuring software development problems*, Addison-Wesley, 2001.