D4.7 Validation of DP'ed Business Processes & Verification of Redeployed Business Processes

WP4 – Processes and Methods for Digitally Preserving Business Processes

Delivery Date: 31/03/2014

Dissemination Level: Public

TIMBUS is supported by the European Union
under the 7th Framework Programme
for research and technological development and demonstration activities (FP7/2007-2013)
under grant agreement no. 269940

Deliverable Lead		
Name	**Organisation**	**e-mail**
Stefan Pröll	SBA	sproell@sba-research.org

Contributors		
Name	**Organisation**	**e-mail**
Tomasz Miksa	SBA	tmiksa@sba-research.org
Stefan Pröll	SBA	sproell@sba-research.org
Stephan Strodl	SBA	sstrodl@sba-research.org
Rudolf Mayer	SBA	rmayer@sba-research.org
Elisabeth Weigl	SBA	eweigl@sba-research.org
Ricardo Vieira	INESC-ID	rjcv@ist.utl.pt
Sven Euteneuer	SQS	Sven.Euteneuer@sqs.de
Perumal Kuppuudaiyar	INTEL	perumal.kuppuudaiyar@intel.com
Jose Barateiro	LNEC	jbarateiro@lnec.pt

Internal Reviewer		
Name	**Organisation**	**e-mail**
Mykola Galushka	SAP	mykola.galushka@sap.com
Carlos Coutinho	CMS	Carlos.Coutinho@caixamagica.pt

Disclaimer

The information in this document is provided "as is", and no guarantee or warranty is given that the information is fit for any particular purpose. The above referenced consortium members shall have no liability for damages of any kind including without limitation direct, special, indirect, or consequential damages that may result from the use of these materials subject to any liability which is mandatory due to applicable law. Copyright 2014 by SBA, SQS, INESC-ID, LNEC, and INTEL.

Table of Contents

List of Figures

List of Tables

List of Acronyms

API	Application Programming Interface
BP	Business process
DP	Digital Preservation
DIO	Domain Independent Ontology
DSO	Domain Specific Ontology
OWL	Web Ontology Language
QA	Quality Assurance
RDF	Resource Description Framework
SPARQL	Simple Protocol and RDF (Resource Description Framework) Query Language
UML	Unified Modelling Language
VM	Virtual machine
WP	Work Package

1 Executive Summary

This document describes the results of:

- Task 4.6 Process and Method for Validation of Preserved Business Processes,
- Task 4.7 Security and Authorisation Process for Preservation and Redeployment,
- Task 4.8 Process and Method for Redeployed Business Processes Verification,

which are part of WP4 (Processes and Methods for Digitally Preserving Business Processes) of the TIMBUS Project. These tasks are responsible for evaluating the results of the preservation processes provided by Task 4.5 (Digital Preservation Process Engineering), verifying the correctness of the process redeployment and description of the security features of the preserved business process supporting the secure preservation and redeployment. The work described in this deliverable is essential for proving that the TIMBUS processes have been used correctly for preserving or redeploying a process. For this reason the instruments and methods introduced in this deliverable are crucial for evaluating the success of the TIMBUS process framework (TIMBUS Consortium, 2013b) application.

The deliverable describes a set of concepts which used together enable a faithful verification and validation of preserved and redeployed processes. The proposed verification and validation framework (VFramework) has a key role in the process of verification and validation, as it provides the guidelines for successful process verification of preserved processes. It describes the key actions which have to be performed in order to verify any kind of redeployed process. The VFramework is used for gathering data about the original process. The information collected by the VFramework application is stored into the VPlan, which is an ontology for organizing and storing of verification information. Furthermore, the deliverable introduces a proof of concept tool called VHelper, which demonstrates the automation of the verification and validation processes. This deliverable also discusses the usage of SPARQL queries for validation of collected data. Finally, the presented concepts are evaluated against two use cases from WP7 (Open Source Workflows) and WP8 (Civil Engineering data transformation process).

Covering the security aspects of business processes, the deliverable provides an overview of the novel TIMBUS Security Ontology that serves as a formal description framework for security features of business processes. This ontology allows users to describe which security features have been implemented by a business process. Additionally it enables preserving the crucial knowledge of the security infrastructure in an abstract representation which can be automatically retrieved and processed further in order to meet future security requirements. In this deliverable, we show how the security information can be extracted from the original process by domain experts and demonstrate how the knowledge can be described in a fashion that is prepared for the upcoming future. Hence, it allows reacting on changes triggered by technical advancement such as obsolescence which is especially crucial within the context of secure process execution. The presented security ontology can easily be integrated into the existing TIMBUS

Context Model and expand it by introducing security features. For demonstrating the fitness of our approach, we applied the TIMBUS Security Ontology on two use cases and extracted its security features.

2 Introduction

Digital preservation aims to keep information and knowledge accessible and to maintain this availability for future generations to benefit from what was achieved and learned until now. A specific kind of knowledge is contained in business processes, which describe how enterprises pursue their business goals by using information technology systems and organizational activities. TIMBUS aims to preserve such business processes for the long term and fostering the reuse of these business processes in the future in potentially new technological environments. As the nature of the future IT systems and their properties are yet unknown, we need to describe processes as detailed as possible, including all information that might become relevant for the redeployment. The approaches we developed within the TIMBUS Project aim to describe how a business process can be analysed, preserved and redeployed within a new environment to overcome technical changes. The TIMBUS framework (TIMBUS Consortium, 2013b) guides users through the business process preservation sequence and has to be flexible enough for supporting arbitrary business processes while being precise enough to prevent any undesired changes.

No matter how well-engineered the preservation of processes is, it cannot guarantee that all necessary information required to run the process was recorded. Given the complexity of preserving entire systems and processes, we thus need to derive means for reliably verifying whether a process being re-deployed performs correctly according to preservation goals. We need to ensure that not only sufficient information is collected during planning and preserving of the process, but also to confirm that the redeployed process performs according to the expectations of the redeployment scenario.

The verification of redeployed processes is a complex task which may vary in its form due to several factors: the way the processes are specified, the drivers for their preservation, the preservation strategies applied; the reasons for the redeployment, the redeployment environments, etc. However, regardless of these differences, all processes must be verified for measuring the success of the redeployment. Otherwise, there is no guarantee that the process running in the redeployed environment is the one which was meant to be redeployed. Such evidence is crucial in litigation cases when the correctness of the original process, executed at some time in the past, could be questioned, and the only way to check this is to re-run the original process. In such cases, the method for verification of redeployed processes should provide irrefutable evidence that the redeployed process is behaving exactly the same way as the original.

Ontologies allow formalizing the knowledge of a certain domain by specifying the vocabulary to describe concepts, the relationships between concepts and the semantics associated with these relationships. The most cited definition of ontologies within the context of computer science was given by (Gruber, 1995): “An ontology is a specification of a conceptualization”. Since the specification is formal,it can be expressed with in an unambiguous way. This clearness allows translating the knowledge that is contained in such an ontology into a computer interpretable format, such as RDF or OWL. As a result, ontologies can be used to describe arbitrary concepts of any domain in a machine processible way.

An ontology does not only define the used vocabulary and formalises the relationships between the concepts explicitly. It also allows to be filled with instances of the concepts. Utilizing the real world objects and the knowledge represented by the ontology and its relationships allows deriving answers to the questions about the domain. Therefore the knowledge can be shared and reused in different scenarios and existing ontologies can be combined with other knowledge representations. This enables the combination of various domains and describes complex areas of interest.

Ontologies and their expressiveness haven been utilized widely throughout the TIMBUS Project as they enable combining knowledge of diverse domains. For this reason and the seamless integration of the knowledge into the TIMBUS context model, we also use ontologies for describing verification, validation and security details of business processes.

The VPlan is an ontology based concept which is used to store and organize information collected during the VFramework application. It is created when the original process is preserved and is accessed during the redeployment. The VPlan links the requirements expressed by significant properties and metrics with the way they are measured. The VPlan uses the context model for depicting precisely from which process' part the information was captured. Moreover, it includes also capturing processes, which were originally modelled in Archimate[1] and later converted into the ontology in order to document the way the data was collected.

Information has become the key asset for modern enterprises as it contributes a significant value to our society. With the increasing network linkage of our digital devices and the ever growing connectivity of the services offered, the need for information security has developed from a niche topic to a challenge that no serious organization can ignore. The Commission of the European Union has expressed its urge for enabling and establishing a secure cyber infrastructure and will "work towards a coherent EU International cyberspace policy to increase engagement with key international partners and organisations, to mainstream cyber issues" (Commission, Cybersecurity Strategy of the European Union: An Open, Safe and Secure Cyberspace, 2013). A precursory public consultation on "Improving network and information security in the EU"[2] showed that security incidents are increasing (57 percent of respondents had experienced security incidents in 2011 that had a serious impact on their activities) and require immediate action in increasing the security of information systems. The European Commission responded with a "proposal for a directive of the European Parliament and the Council concerning measures to ensure a high common level of network and information security across the Union" (Commission, Proposal for a Directive of the European Parliament and of the Council concerning measures to ensure a high common level of network and information security across the Union, 2013).

Protecting information infrastructures and applications from unintended usage is becoming more and more important as there is a tendency of orchestrating different services and distributed components to achieve a business goal. Services need to transmit sensitive data over insecure channels, provide multiuser access with different roles and permissions, encrypt local files and prevent complete systems from unauthorised

[1] Archimate: http://www.opengroup.org/subjectareas/enterprise/archimate

[2] http://ec.europa.eu/information_society/digital-agenda/actions/infosec-consultation/index_en.htm

access. Therefore information security aims to provide only those users and automated services with access to systems that are entitled to. Furthermore checksums and certificates are used to assess the integrity of data and to detect anomalies within large data sets. Again ontologies can be used for describing the knowledge from the security domain.

The goal of the TIMBUS Security Ontology is to provide knowledge of such security concepts and store this information for the long term. We designed the ontology in a generic way since it should allow the mapping of other domain specific ontologies. This enables domain experts to integrate the knowledge of the security details of a business process into an appropriate security ontology that covers future developments in the area of information security which might not even exist at the point of writing this deliverable. Therefore the knowledge contained in the security ontology remains useful for the long term.

The aim of the TIMBUS security ontology is to describe the security features that have been implemented by a process and associate these with users, abstract roles, files, services or sub processes during all three phases of the TIMBUS life cycle (TIMBUS Consortium, 2013b). This description can be used for protecting sensitive data for the long term and for managing and maintaining data with an appropriate level of security.

This collection of security knowledge associated with a business process serves as an inventory of the security features. Whenever a certain technology gets obsolete, an algorithm broken and a security standard revised, the ontology can be used for finding all critical implementations and replace the security control in question with a current version in order to restore a secured version of a preserved business process.

Hence all methods that limit access to data and information are a potential threat to digital preservation and therefore to the goals of TIMBUS. Information security methods such as encryption add an additional layer of complexity to the already challenging problem of preservation. As every other software, encryption libraries can become obsolete quite quickly and newer versions might not guarantee backward compatibility. Also used encryption algorithms might get insecure and hence lose their purpose. The same is true for authorisation mechanisms and permission systems. Preserving complex systems and sensitive data is a complex task. There is always a trade-off between the complexity of the preservation actions and the level of security that has to be maintained. Therefore TIMBUS also needs tools for removing additional levels of security where they are not needed. This includes for instance the abstraction of individual access roles from actual users into generic roles with reduced complexity. Also methods that allow the removal of encryption or the replacement of potentially complex authentication and authorization with simpler yet sufficiently secure mechanisms need to be supported. This diverse set of requirements demands a flexible solution that is independent from actual implementations, but able to express security requirements that fit into the future scenarios of secure business processes.

The work that is presented in this deliverable is closely connected with the achievements presented on the context model (TIMBUS Consortium, 2013a). Both ontologies allow extending the generic context model by domain specific knowledge of verification and validation information as well as precise descriptions of security principles.

The outcomes of this deliverable will be used by WP7 and WP8 when applying the TIMBUS preservation framework.

3 Verification and Validation

During the course of research in tasks:

- T4.6 Process and Method for Validation of Preserved Business Processes,
- T4.8 Process and Method for Redeployed Business Processes Verification,

we have created a set of concepts which used together enable faithful verification and validation of preserved and redeployed processes. This section presents the developed solution. The description starts by setting up a context of the research in the related work section. Then we provide the solution overview and a detailed description of each of its components in the consecutive sections. Finally, using two use cases from WP7 (TIMBUS Consortium, 2013c) and WP8 (TIMBUS Consortium, 2012) we demonstrate how the proposed solution can be applied to verify and validate processes which differ in the level of formalism of their specifications, as well as in preservation requirements.

3.1 Related work

This section discusses the most important works which relate to verification and validation of preserved business processes. To the best of our knowledge, there is no research done, that would directly address the problem of verification and validation of preserved and redeployed business processes. However, there are standards which deal in general with the matter of verification and validation of software and systems. When developing our solution we took them into account and for this reason we present an overview of them below (see Section 3.1.1). We have also focused on research conducted in the area of digital preservation, in particular on existing frameworks for evaluation of digital preservation effects (see Section3.1.2). This has significantly influenced the design of the VFramework (see Section 3.3). We also had a closer look on the significant properties which are broadly used in the digital preservation community to describe the characteristics of the preserved object and investigated the requirements engineering methods (see Section 3.1.3) to better understand how the significant properties can be decomposed into measurable metrics. We have also conducted a research on existing ontologies (see Section 3.1.4) which could directly address the requirements of the VFramework, but none of the existing could be applied. Finally, we discuss the process modelling languages with a specific focus on ArchiMate (see Section 3.1.4).

3.1.1 Verification and validation standards

In (ISO/IEC 12207:2008: Systems and software engineering - Software life cycle processes, 2008) the life cycle processes for systems and software were defined. The standard "contains processes, activities, and tasks that are to be applied during the acquisition of a software product or service and during the supply, development, operation, maintenance and disposal of software products" (ISO/IEC 12207:2008: Systems and software engineering - Software life cycle processes, 2008). It does not consider the redeployment as a part of the life cycle and hence provides no guidance for the scenario considered by TIMBUS. The standard defines also the *software specific processes* and lists actions which are needed for the *Software Verification*

Process and the *Software Validation Process*. However, these processes belong to the *Software Support Process* category which assists the software implementation process. As a consequence, these processes are highly coupled with the software development, what is not in the scope of our investigations.

The IEEE 1012 standard (IEEE Std 1012 - 2004 IEEE Standard for Software Verification and Validation, 2005) specifies a process for software verification and validation. This process addresses the following software life cycle processes: acquisition, supply, development, operation and maintenance. It is compatible with ISO 12207. It defines tasks, required inputs and outputs to conduct verification and validation of the software at all aforementioned life cycle processes. The verification and validation process for the maintenance process considers migrations to other environments. This overlaps with some of the requirements we set to the framework for verification of redeployed processes (see Section 3.3), i.e. the system is migrated to the other platform when the original system is still available. However, it does not consider the situation when the system or the process is disposed, deposited and redeployed after some time. Furthermore, the standard specifies only a high level list of activities applicable in several maintenance scenarios which are rather focused on verification and validation of the activities performed to keep the system running (e.g. system updates, bug fixing, enhancements to the functionality), rather than on digital preservation scenarios. The VFramework (see Section 3.3) provides more detailed guidance and can be applied to a broader range of digital preservation scenarios.

3.1.2 Digital preservation evaluation methods

In (Guttenbrunner & Rauber, 2012) a conceptual framework for evaluation of emulation results was presented. It was demonstrated in (Guttenbrunner & Rauber, 2012b) that the framework can be successfully applied to evaluate the conformance and performance quality of applications and simple processes redeployed in an emulator. This was demonstrated on case studies in which the framework was used to evaluate the emulation of a video game and an accounting program. The VFramework presented in this deliverable (see Section 3.3) is a refinement of that framework for complex, potentially distributed processes. It provides detailed specification of actions which have to be performed for verification of redeployed processes. The VFramework was published in proceedings of a peer reviewed conference (Miksa et al., 2013).

3.1.3 Significant properties and metrics

In (Dappert & Farquhar, 2009) the notion of significant characteristics and properties of a digital object is discussed. The authors aim to remove ambiguities which arose within the digital preservation community around different concepts including the concept of significant properties. They acknowledge that the most common definition used by the community is: "The characteristics of digital objects that must be preserved over time in order to ensure the continued accessibility, usability, and meaning of the objects, and their capacity to be accepted as evidence of what they purport to record" (Wilson, 2007).

The "characteristics of digital objects", mentioned in the quotation above, can be discovered using requirements engineering methods. One of them is Goal-Oriented Requirements Engineering which is a recognized approach that uses goals for the elicitation, negotiation, documentation, and validation of requirements. Goals are a crucial concept in order to understand the intentions of the stakeholders with respect to the objectives, properties, or usage of the system (Pohl, 2010). The main goal-oriented requirements engineering techniques available are the NFR Framework (Chung & Prado Leite, 2009), the i* (i-star) Framework (Yu, 1997)and the KAOS method (van Lamsweerde, 2009). Each of these standards allows organizing and structuring the requirements as goals and therefore enabling better understanding of the requirements set to the system or a process. Each of these methods could be used to obtain significant properties.

The Goal Question Metric (GQM) allows decomposition of significant properties into measurable metrics. The GQM model is divided in three levels (Basili, Caldiera, & Rombach, 1994): the Conceptual Level (*Goal*), where goals are defined concerning products, processes, or resources; the Operational Level (*Question*), where questions are used to characterize the assessment of a specific goal; and the Quantitative Level (*Metric*), where data is associated with each question in order to answer the question in a quantitative way.

3.1.4 Ontologies

The provenance ontologies seemed a natural candidate which may be used at least as a basis for extension. The authors of (Y. L. Simmhan, 2005) "create taxonomy of data provenance characteristics and apply it to current research efforts in eScience, focusing primarily on scientific workflow approaches". The authors claim that there is no common standard for provenance representation and point out that many workflows systems implement it differently. For this reason we investigated one of the popular workflow systems *Taverna*[3] . It stores provenance data as an ontology using *Janus* (P. Missier, 2010). The information contained in the *Janus* ontology describes execution of a workflow, i.e. data exchanged between workflow elements, timestamps, etc. This information is very useful for modelling of the process instance execution, but does not provide information on the significant properties, metrics or the conditions in which the capturing took place. The Wf4Ever[4] project uses thee *wfprov*[5] ontology which is capable of storing information about the execution and parameters of a workflow, but there is also no information on significant properties or capture processes. Furthermore, both *Janus* and *wfprov* are limited to formally specified processes like workflows. Achieving the functionality of the VPlan (see Section 3.4) by linking any other ontology to the *wfprov* or *Janus* ontologies would not be possible and may lead to semantic inconsistencies between the concepts. None of the existing ontologies was suitable to fully address the requirements of the VFramework (see Section 3.3) and neither was the composition of them. Therefore, we have designed the VPlan from scratch.

[3] Taverna: http://www.taverna.org.uk/

[4] Workflow 4Ever project: http://www.wf4ever-project.org

[5] *wfprov* ontology: http://purl.org/wf4ever/wfprov#

3.1.5 Process modelling

Processes, as an organized set of activities performed to achieve a specific desired outcome, is something that exists in all organizations and might be described and documented in many different ways. To the description of a process using a set of key concepts and relations we typically call process modelling. Modelling enables a common understanding easing the analysis of a process (Aguilar-Sav, 2004). There are several techniques to model processes depending on the pretended analysis such as flow charts, data flows, role activity diagrams, etc. (Aguilar-Sav, 2004). Nowadays, the most known and used technique and language to describe the flow of a business process is the Business Process Modelling Notation (BPMN) ((OMG), 2011). Enterprise Architecture (EA) is a coherent set of principles, methods and models to design, analyse, change and manage organizations through four main architecture domains: business, data, application and technology. However, in order to proper describe the main concepts of EA and the dependencies between domains BPMN is insufficient (Susanne Glissman, 2009). Therefore EA languages emerged in order to address the existing gap. ArchiMate (Haren, 2012) represents the culmination of years of work in the area of EA modelling languages and frameworks and is one of the most used EA languages nowadays. It provides high-level abstract concepts divided into three tightly connected EA layers: the business layer, the application layer, and the technology layer. It is a mature language with extensive use and practice where elements and relationships are clearly defined and explained (Susanne Glissman, 2009). By those reasons, in the VPlan presented in this paper Archimate is used to model the required processes namely the preserved process, the capture processes and, if exists, the determinism transformation processes.

3.2 Solution Overview

In this section we present an overview of the solution allowing for verification and validation of preserved business process. We introduce key components and explain the relation between them before providing more details about each concept.

The proposed solution aims at guiding the preservation expert through the verification process. The components are supposed to facilitate verification and validation starting from the data collection and its organization, through its validation, ending up on data presentation. The components and their relations are presented in Figure 1.

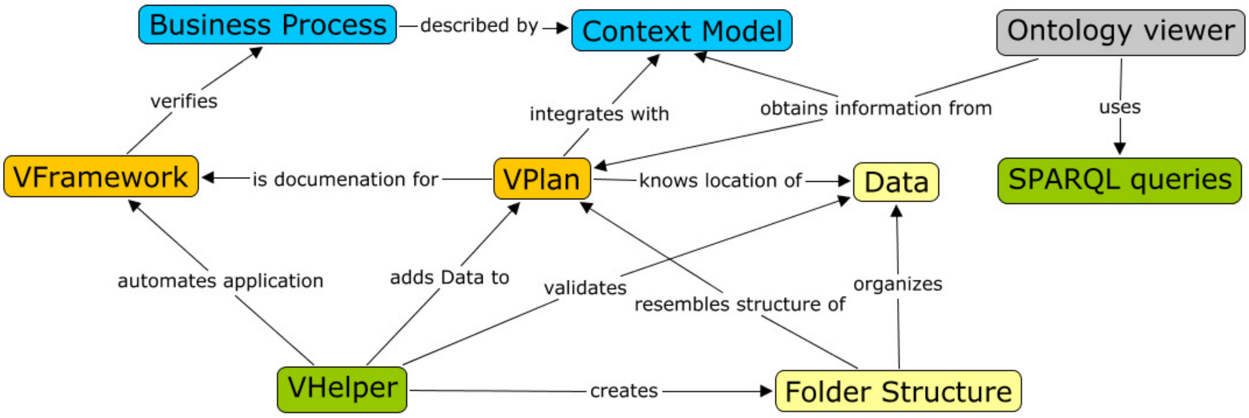
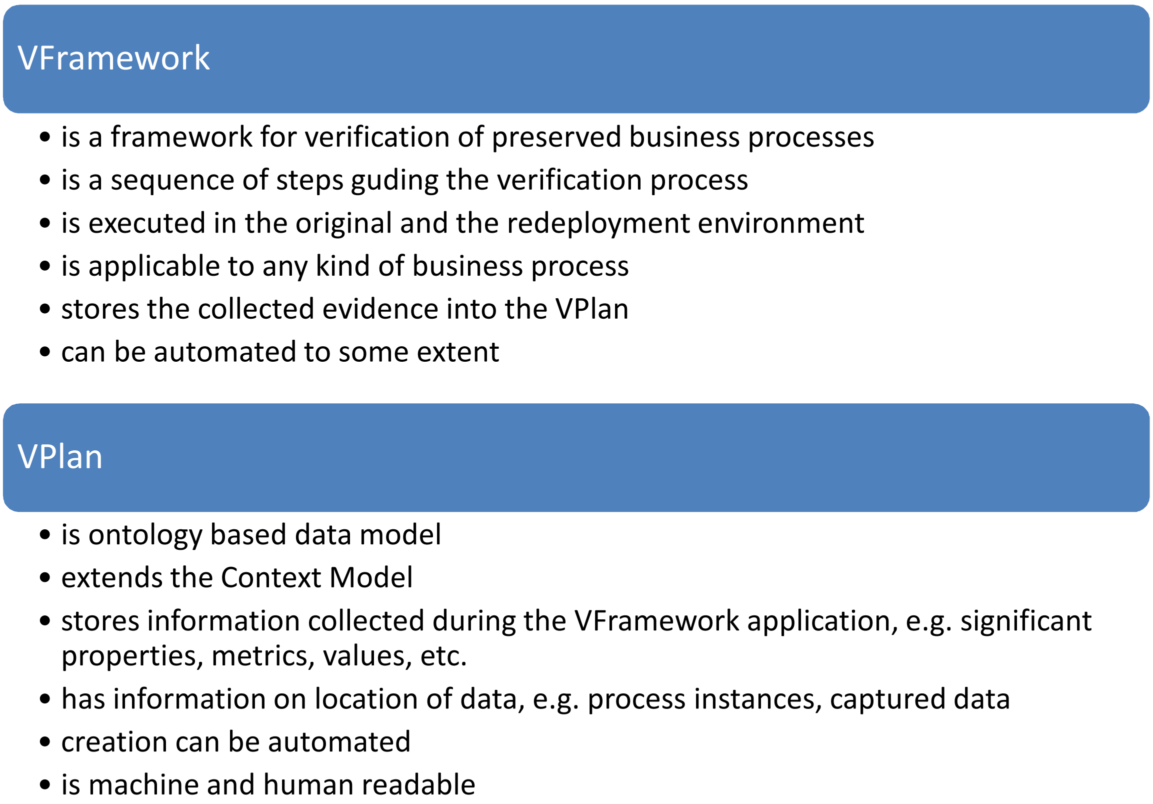


Figure 1: Solution overview concept map. The elements presented in orange are the core concepts of this task; the blue elements are concepts used in TIMBUS; the green elements are proofs of concept developed within this task; the yellow elements are data objects; the grey element is publically available software.

The key components are the VFramework (see Section 3.3) and the VPlan (see Section 3.4). The VFramework is a sequence of steps guiding the verification process. It is executed in the original and the redeployment environment and is applicable to any kind of business process. The VPlan is an ontology based data model which organizes and stores the information collected during the VFramework application. Once the process has been preserved, the VPlan is the ultimate source of information regarding the process metrics. It also provides auxiliary information facilitating verification of redeployed business processes. Only if the captured data used for verification of the process is correctly organized and managed, the verification is possible. Therefore there is a strong binding between the data model (VPlan) and the data, i.e. the VPlan knows the location of data, and the data is organized in a precisely defined folder structure (see Section 3.5). The process of generating the folder structure, adding the information on data location to the VPlan and also validation of the data objects, can be automated. The VHelper is a proof of concept tool that performs these things (see Section 3.5). Similar to the DSOs, the VPlan is mapped and thus integrated with the Context Model (TIMBUS Consortium, 2013a). It complements the description of the preserved process by provision of information on process instances, metrics and data captured for each of them. It can also be queried using SPARQL queries. Such queries can be used for creating reports which deliver necessary information during the redeployment phase, but also for validation of completeness of the VPlan (see Section 3.4). Figure 2 presents an overview of the key features of the VFramework and the VPlan.



VFramework

- is a framework for verification of preserved business processes
- is a sequence of steps guding the verification process
- is executed in the original and the redeployment environment
- is applicable to any kind of business process
- stores the collected evidence into the VPlan
- can be automated to some extent

VPlan

- is ontology based data model
- extends the Context Model
- stores information collected during the VFramework application, e.g. significant properties, metrics, values, etc.
- has information on location of data, e.g. process instances, captured data
- creation can be automated
- is machine and human readable

Figure 2: Overview of key features of the VFramework and the VPlan.

3.3 VFramework

The VFramework was created to verify that a redeployed process performs according to expectations. The framework’s foundation is driven by two major requirements.

Firstly, the framework has to be independent of the situation in which different digital preservation actions were applied to the full process or to different parts of the process. In such situations some of the process' parts may be substituted, re-engineered, emulated, migrated, etc. As a result, the redeployed process which is to be verified is not necessarily an exact copy of the original process.

The framework has to be capable of verifying the execution of similar processes or their parts. By similarity of processes we mean a situation, in which the functionality or characteristics of the process have been altered, but the deviation is either desired (e.g. faster computation) or acceptable (e.g. some functionality is limited but for the purpose of redeployment it is not required). Such situations may be an inevitable side effect of the digital preservation actions or a consequence of deliberate actions (e.g. improved implementation of the process). The framework has to support such situations regardless of its origin, and be capable of evaluating full and partial redeployments of processes.

Secondly, due to the high variety of the nature and implementation of the processes and a wide range of potential user requirements that had to be considered, the framework has to be flexible to cover all these requirements and settings. Therefore it has to remain at a relatively high level of abstraction and be customizable for the concrete processes which are going to be preserved. The guidance on customization is provided by the framework in order to achieve the comprehensiveness of the process verification.

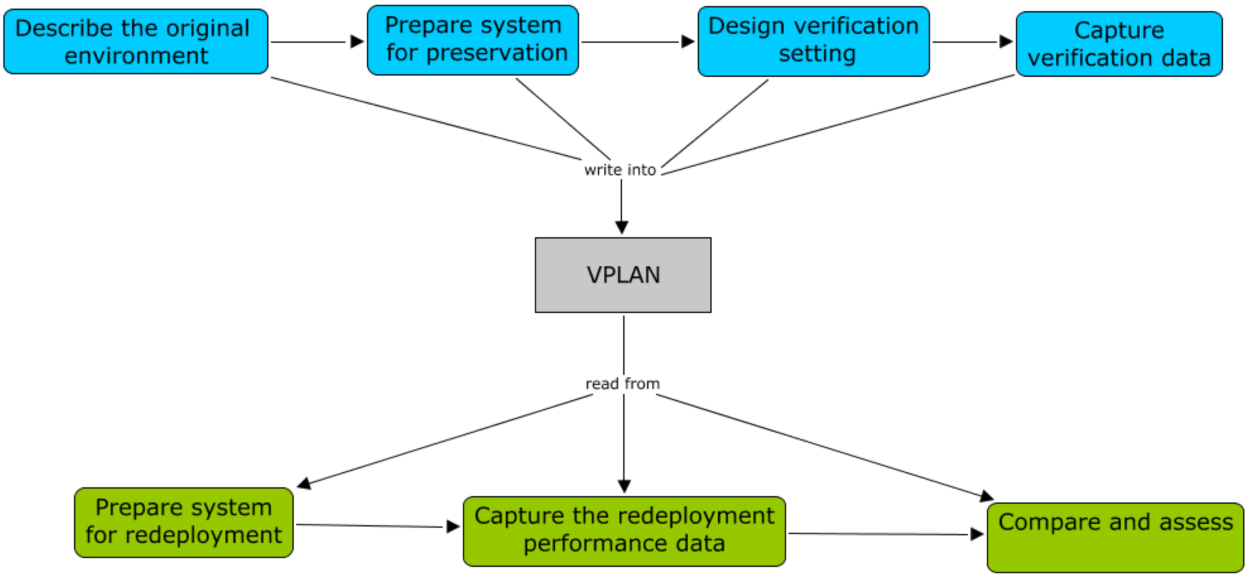

Describe the original environment
Prepare system for preservation
Design verification setting
Capture verification data
write into
VPLAN
read from
Prepare system for redeployment
Capture the redeployment performance data
Compare and assess

Figure 3: VFramework.

The VFramework is depicted in Figure 3 and consists of two sequences of actions. The first one (depicted in blue) is performed in the original environment. The result obtained from execution of each step is stored in the VPlan (see Section 3.4). The second sequence (depicted in green) is performed in the redeployment environment. The necessary information for each of the steps is obtained from the VPlan.

Original environment denotes the system in which the process that is going to be preserved, is deployed and operates. The redeployment environment is the system in which the process will be installed once the decision to redeploy the preserved process is made. It is very likely, that the redeployment will take place at some distant time in the future, when the original platform does not exist anymore and the process may need to be re-engineered to fit it into a new system.

Apart from descriptive metadata, the VFramework uses two kinds of data: verification data and redeployment performance data. The verification data is collected during the execution of the process in the original environment. It provides information on details of the execution of process instances, focusing on measuring significant properties. Interactions with external components have to be stored as well. For this purpose, external interaction data being part of verification data is collected. This external interaction data represents a record of all interactions of the process with external components during the execution of a specific process instance in a scenario to be used for verification. This data is reapplied in the redeployment environment to ensure determinism, by recreating the same external interactions. The redeployment performance data is collected during the execution of the process in the redeployment environment. It provides information on details of the execution of the process instances, focusing on measuring significant properties. It is used for comparison with verification data to assess the redeployment. The steps of the framework are described below and in (Miksa et al., 2013).

3.3.1 Original environment

VFramework step 1: Describe the original environment

The aim of this step is to describe the process and document its context by identifying environment dependencies in which the process is deployed. Information on:

- the motivation for the preservation of the process considered,
- the redeployment scenarios
- set of example instances to be used for verification,

is collected. This corresponds largely to the steps 1-3 of the "Define Requirements" phase in preservation planning (Becker et al., 2008), with the first step being subdivided into two more fine-grained steps.

VFramework step 1.1: Describe the process

The information should describe the process itself but also the context in which the original process operated. A detailed description of not only software and hardware requirements, but also legal aspects is needed. Such information can be provided in multiple forms. The preferable solution is the TIMBUS context model .

VFramework step 1.2: Define set of potential redeployment scenarios

The purpose of the redeployment has to be defined. This information has significant impact on the process of verification, because it influences the type of measurements and the results they are supposed to fulfil. For example, different requirements are set to a process which is supposed to be an exact copy of the original process redeployed for a purpose of litigation case when the correctness of the original process has to be proven and therefore the redeployed process is verified for being identical. Different requirements are set to an eScience process which is redeployed with some of its components substituted with other components of the same functionality but improved quality (e.g. faster computation, more accurate results, etc.). In such cases some of the measurements may be ignored or interpreted differently, e.g. accuracy of results should not be worse than the original, but does not need to be exact. Verification focuses in this case on ensuring the functionality is achieved, but the significant properties related to the part where the changes were introduced should be treated differently.

VFramework step 1.3: Select process instances to be used for verification

A process may have several execution paths and therefore instances of the same process may vary considerably. In this step, the instances of the process which will be used for verification are selected according to the considered redeployment scenarios. The instances selected will be used to collect both verification data from the original environment, as well as the performance redeployment data. The description of selected instances should specify, in a comprehensive way, all actions which were performed when running the process. These could be depicted by sequence diagrams, activity diagrams, use case diagrams, textual description, etc. The form of specification depends on the level of automation of the process, e.g. whether it is a manually executed process or formally specified in BPMN executed within a workflow engine. Furthermore, the values of all parameters and input values must be documented.

VFramework step 1.4: Identify significant properties to be preserved

The significant properties that have to be preserved and then evaluated have to be specified. They can either be collected at this step or obtained from preceding activities, e.g. preservation planning. However, regardless of the source, it is important that the significant properties reflect both functional and non-functional requirements of the process. It is important to determine which significant states of the object are to be measured as the significant property. These significant states could be: target state, continuous stream or series of states (Guttenbrunner & Rauber, A Measurement Framework for Evaluating Emulators for Digital Preservation, 2012).

VFramework step 2: Prepare system for preservation

The aim of this step is to identify the interactions of the process, i.e. all inputs and outputs of the process, but also configurations of process parameters, as well as influences of other components sharing the process environment or used indirectly by process components. This information is needed in order to ensure deterministic execution of the process and thus ensure reliable assessment. The steps should be conducted in view of redeployment scenarios and significant properties defined for the process.

VFramework step 2.1: Determine process boundaries

The process boundary (see Figure 4) specifies which elements belong to the process and which elements belong to the external environment in which the process operates. It is possible to define different process boundaries depending on the scenarios for redeployment. For example, if the scenario assumes redeployment of only a part of the process which will be fitted into another process, then only the redeployed parts of the original process are within the boundary. However, there may be a second scenario in which the full process is redeployed, and then a second boundary has to be defined which covers the entire process. Boundaries may also be influenced by the degree of control one can exert on specific components (e.g. external web services) and their importance for redeployment as well as their stability. In all cases, the description should ensure that the process boundaries are specified clearly, i.e. a distinction between elements which are part of the process to be preserved and which are external services with which the process exchanges data has to be made. This is particularly important in case of distributed processes which are using the Service Oriented Architecture for their implementation, or those deployed in the Cloud.

VFramework step 2.2: Determine external interactions

For each of the specified boundaries the external interactions have to be identified. External interactions denote situations in which elements within the process boundary interact with elements from outside of the boundary. External interactions may be critical for the correct execution of the entire process, because any changes in the external components may cause changes in process execution. For example, the web service which provides data for one of the process steps may change (change of interface, implementation of algorithms, etc.) or become unavailable (Miksa, Mayer, & Rauber, Ensuring Sustainability of Web Services Dependent Processes, 2013). As a result, the process can perform differently (providing different outputs) or cannot run anymore. Another example could be encryption and the necessity to access authentication service. When the certificate is not available anymore, then the communication cannot take place unless the authentication is removed (if the redeployment scenario allows this).

Special attention has to be paid to indirect external interactions and consequences for the process which might not always be visible at the first sight. For example the operating system if not included within the process boundary, its version and all system updates may alter the execution of the process. For all the requirements which focus on the visual presentation, the installed fonts, appearance settings, colour schemes of the system may be such influencers. Other digital objects which coexist in the system may also have impact. For example, processes running in the background (e.g. virus scan software, remote desktop software) can significantly affect the performance of a system. Moreover, other processes may share common data with the examined process and may modify the data that may result in the non-deterministic execution of the analysed process. Furthermore, all user or system I/O (e.g. keyboard, network, specific hardware components such as system clock, etc.) that are outside the process boundaries need to be identified.

VFramework step 2.3: Determine internal interactions

The process may consist of several components which have their own settings. All these settings must be determined at this step. Furthermore, some of the process components depend on further software tools or libraries which may vary in version or settings. Some examples of these could be: virtual machines, database software, libraries, software device drivers, fonts, codecs, etc. The detected versions of components have to be verified to detect if the original versions have not been modified or customized. If some of them were modified (e.g. modified config files) and this has an impact on the process, then they have to be preserved as well. Besides the software dependencies, the underlying hardware has to be considered when searching for potential internal interactions. The process may depend on some proprietary and unique hardware equipment or the underlying hardware may have some specific implementations of algorithms affecting the results obtained in the process. For example, some of the hardware bugs may affect the results delivered by the process. These results will only be achievable on a particular hardware platform (e.g. well-known Pentium FDIV bug had an impact on the results of floating point calculations, and therefore could alter the results of the whole process upon correct redeployment).

VFramework step 2.4: Ensure deterministic behaviour

To allow verification of redeployment we need to ensure that a process performs deterministic. Thus, all interactions identified in 2.2 and 2.3 need to be verified for completeness to ensure deterministic re-execution. If this is not possible within the generic process, adaptations have to be made specifically for verification. If the determinism cannot be ensured, the verification of processes is not very likely to be possible. The investigation of determinism of the process should be conducted in view of considered redeployment purposes. In some settings, some of the non-deterministic influencers are affecting measures which are not important for the purpose of the redeployment. For example, when the exact execution speed is not considered a significant property, then all of the non-deterministic influencers regarding this particular criterion do not have to be considered.

When one of the process steps exchanges data with some third party component (external interaction), the communication can be recorded and replayed in the redeployment environment. If the process depends on the component which affects determinism of the process, it may be possible to substitute the component with a mock-up which does not have this deficiency. An example of such a solution for web services can be found in (Miksa, Mayer, & Rauber, Ensuring Sustainability of Web Services Dependent Processes, 2013), if one of the steps of the non-deterministic process depends on a random number generator, then it may be substituted with a mock-up which always provides the same sequence of values as the one recorded and thus the process becomes deterministic. Of course, such changes to the process must be documented and possibly reverted after the verification process is finished in the actual redeployment, but for the purpose of verification they should be present.

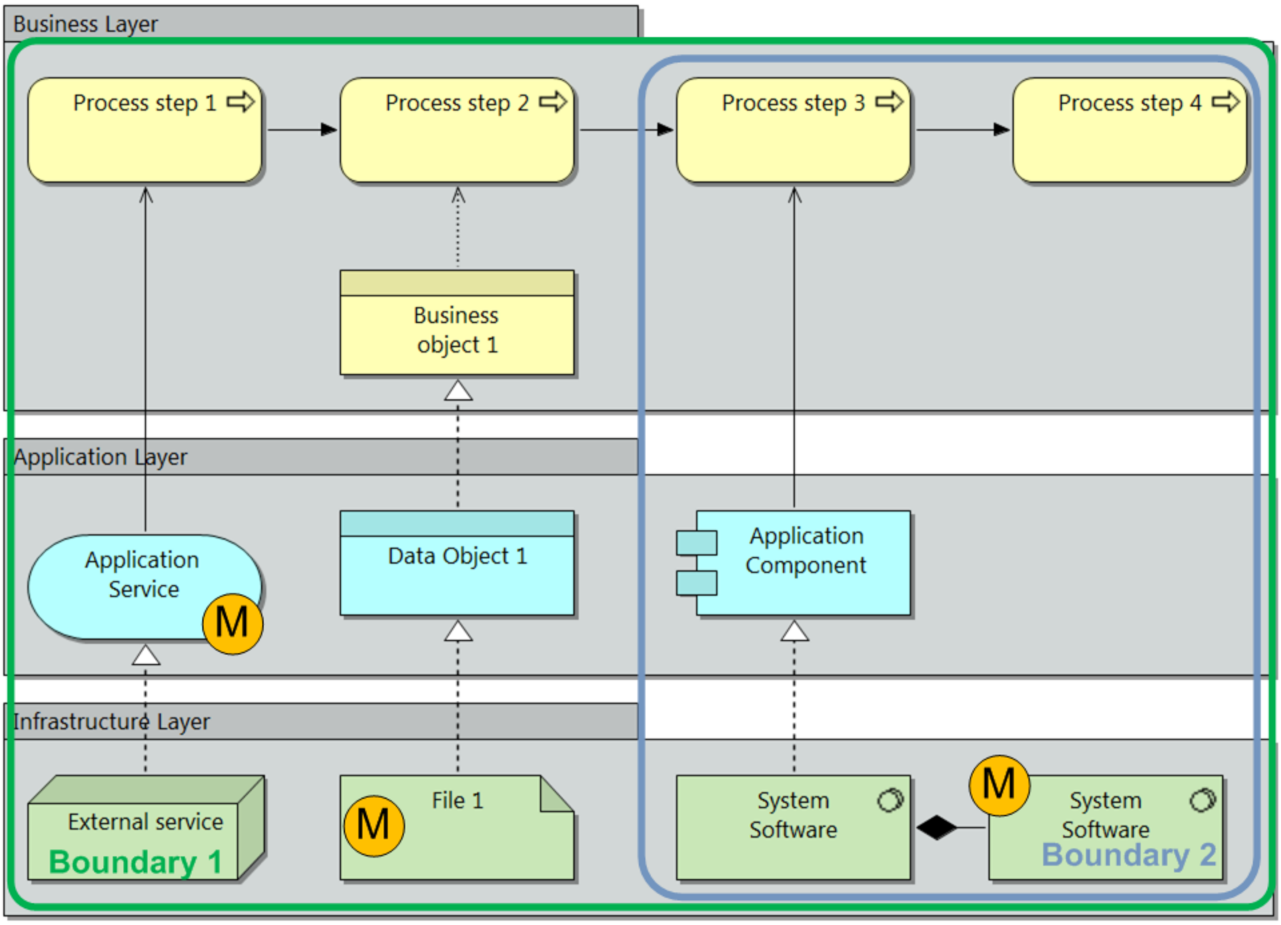


Figure 4: Example of a process modelled in ArchiMate depicting basic concepts of the VFramework. There are two boundaries (green and blue line marking elements belonging to each of them) and three measurement points (orange circles with letter "M" inside).

VFramework step 3: Design verification setting

The aim of this step is to identify the measurement points of the process, specify metrics used to assess quality of preservation actions and couple them with thresholds which are used as criteria for the assessment. The measurement points can be defined as points of the process where data enabling reasoning about correctness of the process execution is collected. The investigation should be conducted in the view of redeployment scenarios and significant properties defined for the process.

VFramework step 3.1: Specify measurement points

Measurement points (see Figure 4) for both internal and external interactions must be described unambiguously and precisely, because the given value can be measured in different ways and in different parts of the process and therefore not always the same values may be obtained. For example, the output of a process that transforms some images into PNG files is selected as a measurement point. This seems to be a clear requirement but without explicit definition of what is exactly measured the results may vary, because the bit streams which write the PNG file to the disk can be compared on the fly or the files already written to the disk can be opened and analysed by image recognition algorithms. In the first case, different libraries may have been used to transform the image (e.g. library was replaced in the redeployment) and as a result the outputs may be different at the bit level, while in the case of image recognition algorithms the images may turn out to be identical. Both approaches are valid and can be used. As the example shows, the choice of the measurement point depends on the requirements and intentions of the future redeployment. We thus need to identify, for each significant property of the process, on which level these must be captured. According to (Guttenbrunner & Rauber, A Measurement Framework for Evaluating Emulators for Digital Preservation, 2012), the core levels are:

- bit level file storage,
- the rendering of an internal state in a the system memory,
- memory of an output device (e.g. video card memory (virtualized or real)),
- port communication (e.g. VGA port, network interface, audio port),
- the actual output device (screen, speakers, actuator).

If the verification aims to check if the rendering algorithms are exactly the same, then the bit comparison seems to be a better measurement point. But if it is allowed to modify the process and only the final visible product needs to be verified, then the second approach should be selected. It may be advisable to take measurements at multiple measurement points and collect the data for all of them. The choice of the measurement point which is most accurate for the redeployment environment will be left to the person redeploying the process who is aware of the reasons and requirements set to the redeployed process.

While measurement points will usually relate to external interactions (e.g. result storage, communication with user or external system), internal interactions within process may be useful to capture for partial redeployments, to allow application and verification of a wider range of preservation actions (such as component replacement) and to allow more flexible redefinition of the boundaries identified in step 2.1. They can also allow a more detailed analysis of where changes originated in case the redeployed process behaves differently.

VFramework step 3.2: Specify metrics for preservation quality comparison

The significant properties which were selected in the first step have to be decomposed from high level significant properties into tangible and measurable metrics which can be measured and identified directly in the process. A wide range of techniques can be used for decomposition. Especially techniques stemming from requirements engineering may be particularly useful in this step, e.g. the above mentioned goal modelling (Young, 2004), GQM method (Basili, Caldiera, & Rombach, 1994). It is also advisable to specify metrics which can identify what the process should not do. In many cases it is easier and quicker to identify the forbidden behaviour or an incorrect state of the process. Then the redeployment can be rejected without a necessity of checking other metrics.

Having defined the metrics, the target values are assigned. These values will be used as the criteria for the assessment. They have to be specified in view of considered purposes of the redeployment. This information has significant impact on the process of verification, because it impacts the importance of available metrics and results they are supposed to achieve. Target values itself can be specified in different ways, e.g. metric A equals Y, metric B is maximum 120% of the original value, etc.

VFramework step 3.3: Aim for automated measurements capture

When the VFramework is applied during planning of the preservation activities and different preservation scenarios and activities are considered, the possibility to automate measurements decreases the time needed for evaluation of alternative preservation strategies. This has lower importance when the VFramework is used during the preservation phase and redeployment phase, when the preservation strategies are already defined. Regardless of the phase, automation of measurements eases the process of verification.

VFramework step 4: Capture verification data

This step has two main tasks. Firstly, to configure the capturing environment for collection of verification data. Secondly, to collect the verification data while the process is monitored by tools which trace process interactions.

VFramework step 4.1: Prepare system for capturing

In this step the capture environment is configured. Either a clean environment is created in which the process is deployed, or an existing instance of an operational system is used directly.

VFramework step 4.2: Prepare data capture tools

Tools for capturing external interactions, as well as verification data are introduced to the capture environment in the next two steps.

VFramework step 4.2.1: Set up tools for capturing external interactions

Tools which will intercept external interactions of the process are installed in the capture environment. The captured information will be used to ensure deterministic execution of sample process instances (step 1) in the redeployment environment.

VFramework step 4.2.2: Set up tools for capturing verification data

Tools which collect data in previously specified measurement points are installed in the capture environment. The captured information will be used to evaluate performance of the redeployed process.

VFramework step 4.3: Run the process and capture data

When the capture environment has been configured and the tools for capturing data are in place, the instances of the process, which were identified in the first step, are executed. The data is being collected during and after the execution of the process.

VFramework step 4.4: Verify validity of captured data

Once the execution of process instances has finished, the recorded data is verified for its correctness. This could be either manual or automatic action, which checks if all the measurements were stored correctly, e.g. if the log files are not empty. If all the data is correct then it is stored into the VPlan.

3.3.2 Redeployment environment

VFramework step 5: Prepare system for redeployment

This is the first step performed in the redeployment environment. This step has three main objectives. Firstly, to configure the redeployment environment for collection of redeployment performance data. Secondly, to redeploy the process in a new environment. Thirdly, to execute the process instances.

VFramework step 5.1: Prepare redeployment environment

The environment in which the process will be redeployed has to be selected. Tools which ensure determinism during execution of the process, as well as the tools used for data collection have to be installed.

VFramework step 5.1.1: Set up redeployment system

Especially if the process is run in an environment shared by other process an analysis of possible external interactions has to be conducted in order to ensure that the determinism of the redeployed process is not affected by a new environment. Also the environments in which no other processes are executed should be checked for any sources of possible determinism disruptions.

VFramework step 5.1.2: Set up external interactions replay to ensure determinism

The external interactions data is used in this step to recreate the interactions of the system. Tools which allow replaying of this data have to be installed in the redeployment environment.

VFramework step 5.1.3: Set up data capture tools

Similarly to the step 4.2, the tools which extract redeployment performance data are installed in the redeployment environment. These tools will collect data needed for verification of the redeployed process at the predefined measurement points.

VFramework step 5.2: Redeploy preserved process

The preserved process is redeployed in this step. Required adjustments to run the process in a new environment are done and the instances of the process which were used in the original environment are executed.

VFramework step 5.2.1: Identify required preservation actions to enable redeployment

The aim of this step is to ensure that the process becomes operational in a new environment and that all of the instances of the process defined in the first step can be executed.

It is very likely that the preserved process will have to be re-engineered in order to be fitted into the new environment. For example, in the given environment a certain library responsible for encrypted communication with a web service cannot be used. However, a substitute library which allows communicating with a web service with a different encryption mechanism might be available. Then such substitution has to be made in order to make the process operational (only if the redeployment scenario does not exclude such an action). In this step all kinds of preservation actions such as replacing a library with another one, cross-compiling code, migrating a file, putting an additional wrapper around the component, etc. may be applied.

VFramework step 5.2.2: Re-run the set of process instances

The process instances which were defined in the first step and executed in the original environment to collect verification data are executed in this step in order to create redeployment performance data. The execution is controlled by the tools which ensure determinism of the process.

VFramework step 6: Capture redeployment performance data

The aim of this step is to collect the redeployment performance data from the new system and verify if the data collection conditions were fulfilled.

VFramework step 6.1: Collect redeployment performance data

The redeployment performance data is recorded by the tools which are monitoring the execution of process instances. All this data is collected and will be used for comparison with the verification data.

VFramework step 6.2: Verify validity of captured data

Before the data can be used for comparison, its validity and fulfilment of assumed level of determinism of the environment needs to be checked.

VFramework step 6.2.1: Verify if required level of determinism was reached

Results have to be analysed regarding the required level of determinism in the environment. If it was possible to ensure it and the tools which were introduced for this purpose in the step 5 performed its task correctly then the requirements are fulfilled. Otherwise, the procedure has to be repeated starting from step 5, and new ways of ensuring deterministic execution of the process have to be introduced.

VFramework step 6.2.2: Verify correctness of captured data

Similarly to step 4.4, the collected redeployment performance data needs to be verified before it can be used for further analysis. This could be either manual or automatic action which checks if all measurements were stored correctly, e.g. if the log files are not empty.

VFramework step 7: Compare and assess

The comparison of significant properties measured in both environments is conducted in this step. The comparison is described in a report and a decision about fulfilment of redeployment purposes is made.

VFramework step 7.1: Compare redeployment performance data and verification data

In this step the comparison between the verification data and the redeployment performance data is conducted. The comparison has to be done by contrasting the data collected at each of the measurement points of the original process with the data collected at each of the measurement points of the redeployed process. Due to the changes which might have been introduced to the process by preservation actions, some of the measurement points may not be available. If so, the comparison is either omitted or another corresponding point is used.

VFramework step 7.2: Conduct preservation quality comparison

The metrics which were specified in Step 3.2 are calculated for the redeployed process. These metrics allow assessing the quality of preservation actions. These metrics are always interpreted depending on the redeployment scenario, because they may have different target values depending on the scenario. In some scenarios a specific functional or non-functional metric may need to be fulfilled, while in another scenario it may not be a requirement.

VFramework step 7.3: Provide summary report

A report summarising the comparison is created. The report is supposed to deliver credible information about the state of the redeployed process; measurements made metrics and their expected values and any alterations detected which are not compliant with the purpose of the redeployment.

VFramework step 7.4: Make the final decision

The final decision is made by the preservation expert who knows the reason for the redeployment and using the report can make a credible decision.

VFramework step 7.5: If positive, remove tools used for verification

If the process is positively evaluated, then the tools for ensuring determinism are removed from the environment, unless they are needed for the redeployed process execution. The original implementations or substitute services providing the full functionality are used instead. Similarly the tools for data collection can be removed from the environment.

3.4 VPlan

The VPlan is an ontology for storing and organizing information collected by the VFramework application. The consecutive subsections describe its structure, the integration with the context model and ArchiMate; classes and properties; as well as mapping to the VFramework steps.

3.4.1 Overview

The VPlan is created when the original process is preserved and accessed during the redeployment phase. It was designed to handle the information collected by the VFramework application. A VPlan is created per process and it contains process instances which can verify particular process execution. The VPlan is publically available at https://timbus.teco.edu/svn/public/ontologies/VPlan.owl.

Figure 5 depicts the concept map of the VPlan. The light blue boxes are the classes, e.g. *VPlan*, *Metric*, *RedeploymentScenario*, etc. The named arrows connecting the light blue boxes are object properties relating classes to each other, e.g. *measures*, *appliesToScenario*, *hasInstance*, etc. The arrows which point to the green boxes are data properties, these are namely: *isLocatedAt*, *hasTextDescription* and *isInline*. There are also five dark blue boxes, which are individuals used for creating an enumeration for the *MetricTargetOperator* class. Finally, there are 3 grey boxes which depict elements imported to the VPlan by importing the Context Model DIO (see Section 3.4.2).

In general the VPlan links the requirements expressed by significant properties and metrics with the way they are measured. To describe the measurement process the information on process instances and capturing processes is provided. The VPlan uses the context model to precisely depict from which process' part the information was captured. Moreover, it includes also the capturing processes, which were originally modelled in Archi and later converted to an OWL ontology in order to document the way the data was collected. Finally, the VPlan stores not only information on data location used to run the process, but also the data which was captured from the process.

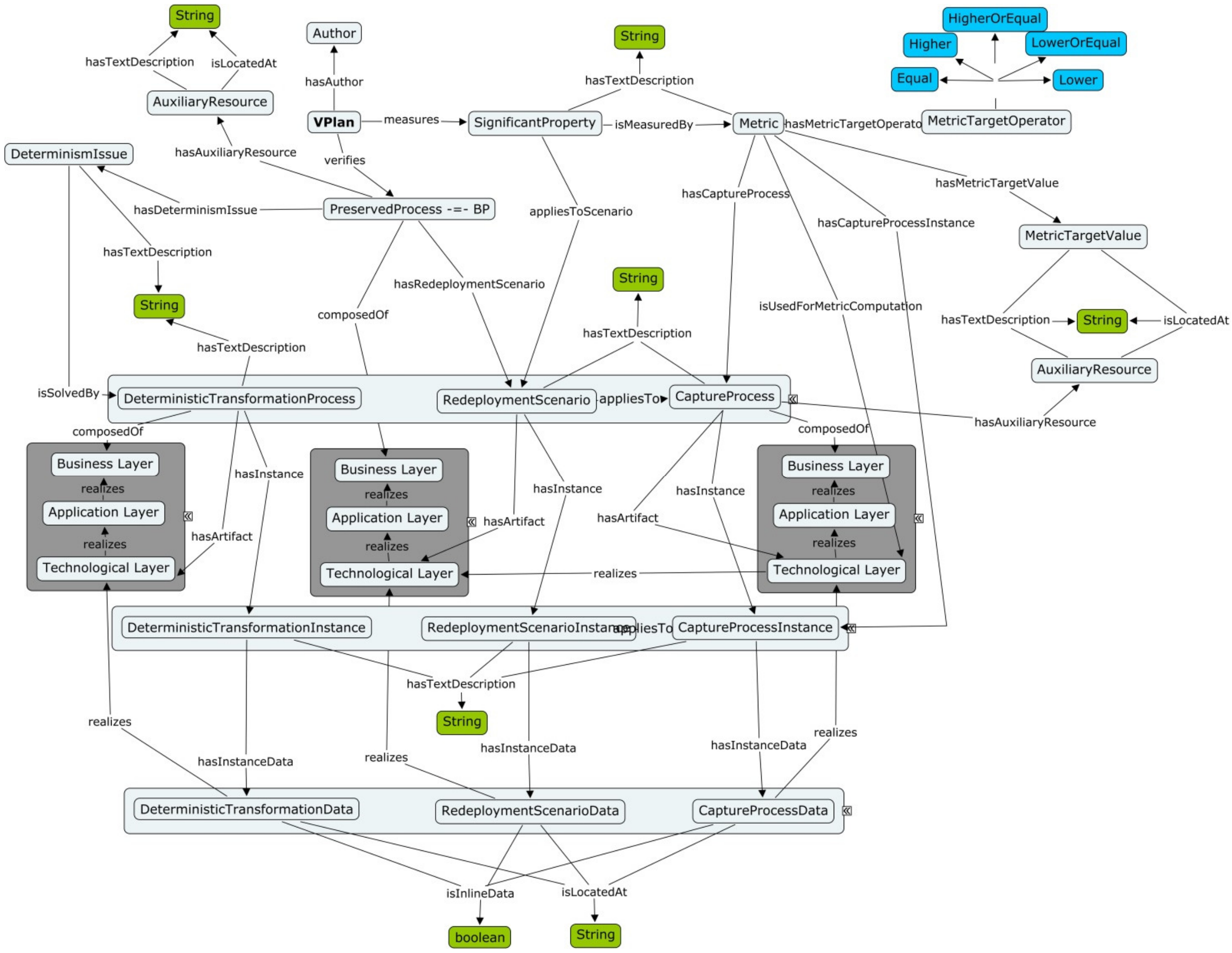


Figure 5: VPlan concept map depicting class, object and data properties.

3.4.2 Relation to the Context Model

Due to the fact, that the VPlan is an OWL document it benefits from integration with other ontologies. By default it is integrated with the DIO developed by TIMBUS. Furthermore, if different concepts are needed, the VPlan can integrate with any other existing ontology. The VPlan uses the context model (DIO) in four different ways which are:

- import of DIO concepts at the model level,
- import of preserved process at the instance level,
- import of capture process at the instance level,
- import of determinism transformation process at the instance level.

Figure 6 illustrates relation of the VPlan to the context model. Each of the cases is discussed in the consecutive subsections.

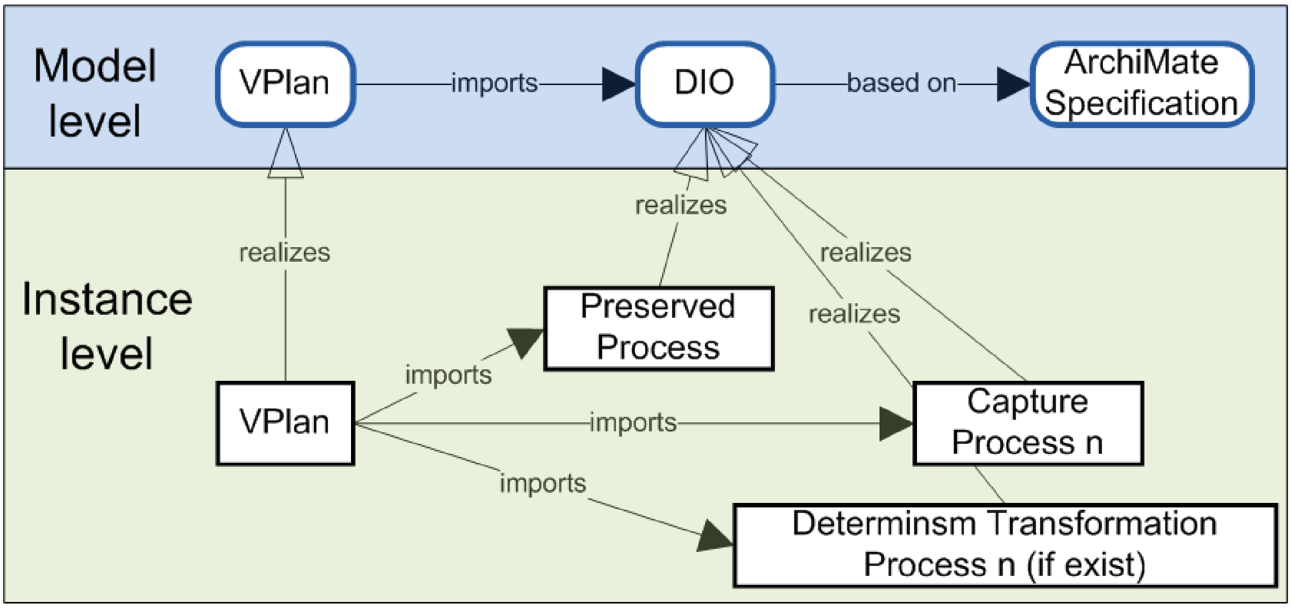


Figure 6: Differentiation between the VPlan model and the instance and an overview of imports made to the VPlan.

3.4.2.1 Import of DIO concepts at the model level

The VPlan is coupled with the DIO at the model level and can thus make an extensive use of the machine readable representation of the process. Moreover, the DIO is based on ArchiMate which is a recognized standard by many Enterprise Architects. Therefore reuse of concepts from the DIO in the VPlan facilitates VPlan understanding to users from those communities.

The VPlan uses the imported DIO in two ways:

- defines some of the VPlan and the DIO classes equivalent,
- reuses some of the object properties from the DIO to link VPlan classes.

In the first case the VPlan, for example, defines the *VPlanProcess* and the *BusinessProcess* to be equivalent. This means that semantically these two concepts are the same.

In the second case, for example, the *realizes* object property is used to link the *VPlanData* to the *Artifact* from the DIO. For both cases, the semantic meaning is much clearer to the preservation expert, because it has exactly the same definition as the ArchiMate standard.

3.4.2.2 Import of Preserved Process at the instance level

The TIMBUS preservation process assumes that in one of the initial steps a context model of the preserved process is created. Because the VPlan is always targeted at a particular process, then a coupling of the VPlan and the context model of the preserved process is natural. This is achieved by importing the ontology based representation of the process into the instance of the VPlan (see Figure 6). As a result it is possible to:

- define redeployment scenarios,
- specify measurement points and levels of comparison.

In case of redeployment scenarios (see Section 3.3.1) the scenario can be expressed by connecting *RedeploymentScenario* individual with each process step of the preserved process. As a consequence, further dependencies of each process' step can be inferred automatically without the need of explicit specification. When it comes to the specification of measurement points (see Section 3.3.1), they can be pointed directly in the preserved process and thus any ambiguities, which could stem from verbal description, are removed. The levels of comparison (see Section 3.3.1) are implicit and depend on the kind of the process element to which the measurement point links.

3.4.2.3 Import of Capture Processes at the instance level

The VPlan requires that for each of the metrics a capture process is defined which describes how the data, which is later used for metric computation, is extracted from the process. An approach similar to the one in Section 3.4.2.2 regarding the import of the preserved process model was taken. Thus each capture process is firstly modelled in ArchiMate, then converted to an OWL ontology and finally imported to the VPlan (see Figure 7).

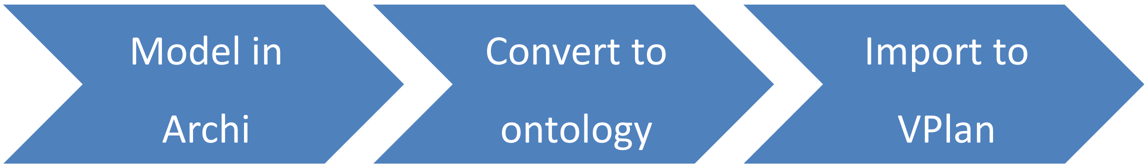


Figure 7: Simplified process of adding Preserved Process and Capture Processes to the VPlan.

Import of the capture process into the VPlan allows linking of the elements of the capture process with the elements of the preserved process. The link is essential, because in this way the generic process of capturing becomes concrete for the given preserved process. In other words this link specifies the measurement point (see Section 3.3.1). For example, most of the capture processes provide at their output a file with some data extracted from the process. In order to state from which part of the process and at which component the capturing took place, the link between the *CaptureProcess* and the *PreservedProcess* is established.

3.4.2.4 Import of Determinism Transformation Process at the instance level

When the process is not deterministic during its execution, i.e. has different characteristic and outputs for the same input data; then it is impossible to conduct faithful verification. The VFramework foresees such a situation and assumes that for the purpose of verification the process part which introduces the lack of determinism can be removed or substituted with a deterministic one. Due to this fact, the VPlan holds information on determinism transformation processes. These processes describe what has to be done in order to make the preserved process deterministic for the purpose of verification. Similar to the capture processes described in the section above, the determinism transformation processes are also modelled in Archi, converted to ontology and then imported to the VPlan (see Figure 7).

3.4.3 Working with the VPlan

The VFramework determines which information at which step is stored into the VPlan. Nevertheless, the aim of this section is to explain how the creation of the VPlan for a particular business process looks like from a more practical point of view.

Figure 8 depicts the general process of VPlan creation. At the beginning the preserved process, capture processes and determinism transformation processes (if exist) are modelled in Archi. Then using Archi to OWL Converter, implement by TIMBUS, the processes are converted into OWL ontologies. Having done this, the preservation expert continues their work in an ontology editor, where they initialize the VPlan instance by starting a new empty ontology. They need to import the VPlan model, converted preserved process, the capture processes and the determinism transformation processes (if exist). Further work consists in creating individuals and adding properties to the VPlan instance.

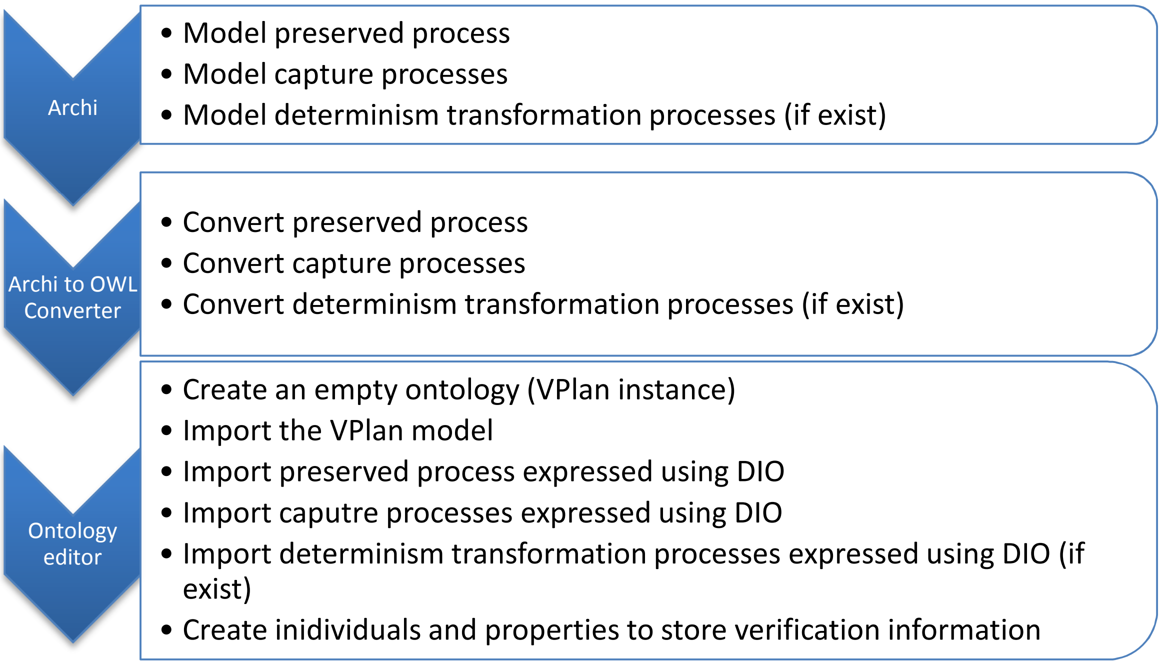
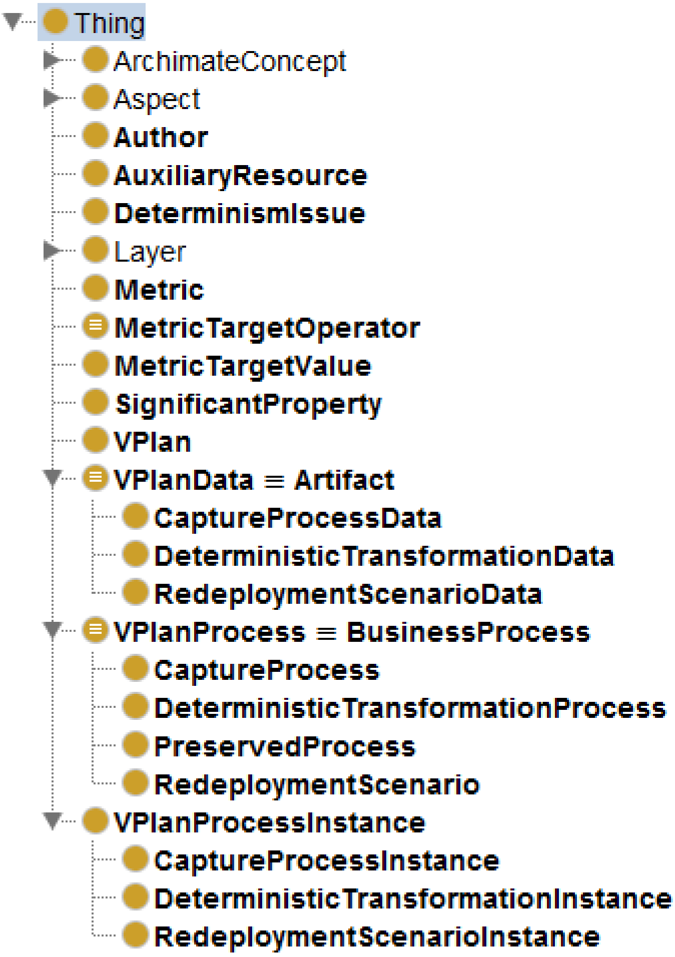

Archi
- Model preserved process
- Model capture processes
- Model determinism transformation processes (if exist)
Archi to OWL Converter
- Convert preserved process
- Convert capture processes
- Convert determinism transformation processes (if exist)
Ontology editor
- Create an empty ontology (VPlan instance)
- Import the VPlan model
- Import preserved process expressed using DIO
- Import caputre processes expressed using DIO
- Import determinism transformation processes expressed using DIO (if exist)
- Create inidividuals and properties to store verification information

Figure 8: VPlan creation process.

One should note that the process of working with the VPlan instance can be facilitated by software tools. VHelper tool described in Section 3.5 allows generating parts of the ontology. Development of further tools which could, for example, provide web interface for provision of data is possible and would very likely ease the process of VPlan creation. However, the process presented here can be performed with a use of currently existing tools.

3.4.4 Classes and their properties

The consecutive subsections describe each of the classes used in the VPlan and also their properties. Figure 9 presents a list and the structure of classes of the VPlan. The structure of the ontology was presented in Figure 5.



- Thing
  - ArchimateConcept
  - Aspect
  - Author
  - AuxiliaryResource
  - DeterminismIssue
  - Layer
  - Metric
  - MetricTargetOperator
  - MetricTargetValue
  - SignificantProperty
  - VPlan
  - VPlanData ≡ Artifact
    - CaptureProcessData
    - DeterministicTransformationData
    - RedeploymentScenarioData
  - VPlanProcess ≡ BusinessProcess
    - CaptureProcess
    - DeterministicTransformationProcess
    - PreservedProcess
    - RedeploymentScenario
  - VPlanProcessInstance
    - CaptureProcessInstance
    - DeterministicTransformationInstance
    - RedeploymentScenarioInstance

Figure 9: The VPlan class hierarchy.

3.4.4.1 Author

Author specifies the author of the VPlan. Information can be provided either in a text form using a *hasTextDescription* data property, or preferably by linking the concept to some external ontology, like for example Friend-Of-A-Friend[6] ontology.

[6] FOAF ontology: http://www.foaf-project.org

3.4.4.2 AuxiliaryResource

AuxiliaryResource is used to provide information on additional resources which may be useful to understand the given item. For example, it could be used for provision of process description. Having a text document or an image of the process model may help in understanding the process.

AuxiliaryResource may have two data properties:

- *hasTextDescription* – to describe what kind of the resource it is,
- *isLocatedAt* – to specify the location of the resource.

3.4.4.3 CaptureProcess

CaptureProcess is used to aggregate the elements of the imported ontology which depicts the data capture process to be used for calculation of metrics. Following properties are used by this class:

- *composedOf* – to specify elements of the imported ontology,
- *hasInstance* – to specify the instances of the *CaptureProcess* (Business Layer elements are sufficient),
- *hasArtifact* – to specify for which elements from the imported capture process ontology the actual data will be stored,
- *hasAuxiliaryResource* – to provide more information about the capture process if necessary.

3.4.4.4 CaptureProcessData

CaptureProcessData is used to provide information about the data collected for the given element of the given *CaptureProcessInstance*. See also *VPlanData*.

3.4.4.5 CaptureProcessInstance

CaptureProcessInstance is used to create a binding between the *CaptureProcess* and *RedeploymentScenarioInstance*. In other words, the *CaptureProcessInstance* is an instance of the *CaptureProcess* for a given instance of the redeployment scenario.

CaptureProcessInstance has two properties:

- *hasInstanceData* – to link the data which was collected for the instance,
- *hasTextDescription* – to provide additional description concerning the instance.

3.4.4.6 DeterminismIssue

DeterminismIssue provides information about detected problems with determinism of the process, e.g. random number generators used.

DeterminismIssue has two properties:

- *hasTextDescription* – to describe what the issue is,

- *isSolvedBy* – to specify how the original process was converted into a deterministic process for taking the measurements.

3.4.4.7 DeterministicTransformationData

DeterministcTransformationData is used to provide information about the data collected for the given element of the given *DeterministcTransformationInstance*. See also *VPlanData*.

3.4.4.8 DeterministicTransformationInstance

DeterministicTransformatinInstance is used to aggregate the data (static data but also tools) which is stored for making the preserved process deterministic

DeterministicTransformatinInstance has two properties:

- *hasInstanceData* – to link the which data was collected for the instance
- *hasTextDescription* – to provide additional description concerning the instance.

3.4.4.9 DeterministicTransformationProcess

DeterministicTransformatinProcess is used to aggregate the elements of the imported ontology, which depicts the determinism transformation process to be conducted for making the preserved process deterministic during the verification. Following object properties are used by this class:

- *composedOf* - to specify elements of the imported ontology,
- *hasInstance* - to specify the instances of the *DeterministicTransformatinProcess*,
- *hasArtifact* - to specify for which elements from the imported capture process ontology the actual data will be stored

3.4.4.10 Metric

Metric is used to decompose and transform significant property into measurable values.

Metric has the following properties:

- *hasMetricTargetOperator* – to specify what is the expected relation between metric values,
- *hasMetricTargetValue* – to specify what the expected value of the metric is during the redeployment; if no value is provided then the data extracted in the original and the redeployed environment (pointed by *isUsedForMetricComputation*) are compared and assed taking into account the *MetricTargetOperator*; otherwise the value captured in the redeployment environment is compared with the value specified by the *MetricTargetValue*,
- *hasCaptureProcessInstance* – to specify for which redeployment scenario instance the data was collected,

- *isUsedForMetricComputation* – to specify which element of the capture process is used for calculation of metric,
- *hasCaptureProcess* – to assign the capture process to the metric.

3.4.4.11 MetricTargetOperator

MetricTargetOperator is used to define relation between value of the metric in the redeployment and the original environment. This class is enumeration, which can take only one of the following values: Equal; Higher; Higher or Equal; Lower or Equal; and Lower.

3.4.4.12 MetricTargetValue

MetricTargetValue is used to specify what the expected value of the metric is during the redeployment. If no value is provided then the data extracted in the original and the redeployed environment (pointed by *isUsedForMetricComputation*) are compared and assed taking into account the *MetricTargetOperator*; otherwise the value captured in the redeployment environment is compared with the value specified by the *MetricTargetValue*.

MetricTargetValue has two data properties:

- *hasTextDescription* – to describe what kind of the resource it is,
- *isLocatedAt* – to specify data location.

3.4.4.13 PreservedProcess

PreservedProcess is used to aggregate elements of the imported ontology depicting preserved process.

PreservedProcess has following properties:

- *hasRedeploymentScenario* – to specify redeployment scenarios,
- *composedOf* – to specify components of the imported ontology (Business Layer elements are sufficient),
- *hasDeterminismIsssue* – to specify if there any determinism problems of the process run,
- *hasAuxiliaryResource* – to provide additional description of the process, e.g. process documentation, UML models, etc.

3.4.4.14 RedeploymentScenario

RedeploymentScenario specifies what scenario for redeployment is considered, e.g. legal obligation, upgraded version of process, etc. This has impact on interpretation of assessment.

RedeploymentScenario has following properties:

- *composedOf* – to specify components of the imported ontology depicting preserved process (Business Layer elements are sufficient),

- *hasInstance* –to specify instances of the redeployment scenario,
- *hasArtifact* – to specify for which process elements the data is collected to build up an instance,
- *appliesTo* – to specify to which *CaptureProcess* it applies.

3.4.4.15 RedeploymentScenarioData

RedeploymentScenarioData is used to provide information about the data collected for the given element of the given *RedeploymentScenarioData*. See also *VPlanData*.

3.4.4.16 RedeploymentScenarioInstance

RedeploymentScenarioInstance is used to define instances of the preserved process. The instances are executed during the verification process. Each instance has *RedeploymentScenarioData* which holds specific data used for executing the instance.

RedeploymentScenarioInstance has following properties:

- *appliesTo* – to build up a *CaptureProcessInstance* together with a *CaptureProcess*,
- *hasInstanceData* – to link the data which was collected for the instance.

3.4.4.17 SignificantProperty

SignificantProperty expresses the verified characteristic/requirements of the preserved process. The assessment of the redeployment is conducted by assertion of fulfilment of the significant properties.

SignificantProperty has following properties:

- *isMeasuredBy* – to specify which metrics are used to determine if the significant property is fulfilled,
- *appliesToScenario* –to specify in which redeployment scenarios the significant property is considered,
- *hasTextDescription* – to express the significant property.

3.4.4.18 VPlan

VPlan is the root of the ontology. It links the preserved process with the significant properties which will be used for verification of the process.

VPlan has following properties:

- *has Author* – to specify the author of the VPlan,
- *measures* – to specify the significant properties used for verification,
- *verifies – to* specify the preserved process verified by the VPlan.

3.4.4.19 VPlanData

VPlanData is used to specify location of the data collected. The data is stored in most cases in a dedicated folder and the *VPlanData* provides a link to this folder. Sometimes the data may be stored directly in the ontology. Then it is called an inline data. The *VPlanData* has three subclasses: *CaptureProcessData*, *DeterministicTransformationData*, *and RedeploymentScenarioData*. The properties of the *VPlanData* class are:

- *isInlineData* – flag to inform whether the data is stored directly in the VPlan (true), or outside the ontology (false),
- *isLocatedAt* – path to the location of data,
- *realizes* – object property defined by DIO, used to depict for which process element the data is stored.

3.4.4.20 VPlanProcess

VPlanProcess is a class which is equivalent to the *BusinessProcess* from the DIO. It is used to group processes modelled in the VPlan. The *VPlanProcess* has following subclasses: *CaptureProcess*, *DeterministicTransformationProcess*, *PreservedProcess*, and *RedeploymentScenario*.

VPlanProcess and also its subclasses have one data property:

- *hasTextDescription* – to provide additional information about the process.

3.4.4.21 VPlanProcessInstance

VPlanProcessInstance is used to group instance of process modelled in the VPlan. It has following subclasses: *CaptureProcessInstance*, *DeterministicTransformationInstance*, and *RedeploymentScenarioInstance*.

VPlanProcessInstance and also its subclasses have one data property:

- *hasTextDescription* – to provide additional information about the process instance.

3.4.5 Mapping to the VFramework

The VPlan was designed to organize and store information collected during the VFramework application. In this section the mapping of the VFramework steps to the VPlan classes is presented. The aim of the mapping is to demonstrate, that the VPlan fulfils the requirements of the VFramework. For this reason, two figures depicting mapping of concepts in the original and in the redeployment environment were created and are discussed in the consecutive subsections.

3.4.5.1 Original environment

The VFramework steps which are executed in the original environment focus on collection of process information. At this phase the VPlan is created and filled with data. The Figure 10 depicts which VPlan classes are used at which step of the VFramework. The numbers on the arrows depict the concrete steps and sub steps of the VFramework. If all sub steps of a given step of the VFramework are making use of a given class, then only a number of a step is provided on the arrow, e.g. *AuxiliaryResource* is used at all sub steps of the *Describe the original* environment step of the VFramework, hence only “1” is used instead of “1.1/2/3/4”.

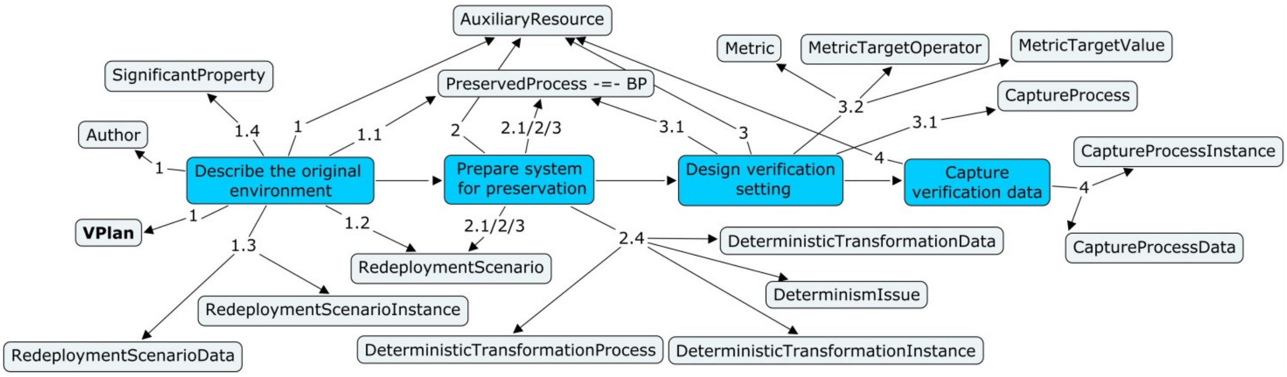


Figure 10: Mapping of the VFramework steps executed in the original environment to the VPlan classes.

In the first step of the VFramework, which is *Describe the original environment*, not only the process and its context is described, but also the redeployment scenarios, verification instances and significant properties. According to the Figure 10 all these concepts are mapped to the respective classes.

In the second step of the VFramework, which is *Prepare system for preservation*, a precise analysis of the process and its dependencies is conducted. Actually, this is the moment when the context model of the process is needed. The internal and external interactions of the process which are identified are modelled in the context model. The process boundaries are defined using *RedeploymentScenario* by specifying steps of the process which belong to the process. The deterministic behaviour is described using *DeterminismIssue* and a way of tackling it with a use of classes related to the transformation process.

In the third step of the VFramework, which is *Design verification setting*, the measurement points are specified by designing capture processes and linking them to the elements of the context model. The metrics for preservation quality comparison have also their respective classes for expressing the metrics and their value.

In the fourth step of the VFramework, which is *Capture verification data*, the data is captured from the process by execution of process instances. The information on data location for each of the instances is also covered by the VPlan.

For more information on classes depicted in Figure 10 see Section 3.4.4.

3.4.5.2 Redeployment environment

The VFramework steps, executed in the redeployment environment, focus on the actual verification of the redeployed process using the information collected in the original environment. At this phase the VPlan is accessed to read the information from it. The Figure 11 depicts which VPlan classes are used at which step of the VFramework. The convention used in the figure is similar to the one from the previous section. The only difference is the direction of the arrows is opposite, since information is read from the VPlan.

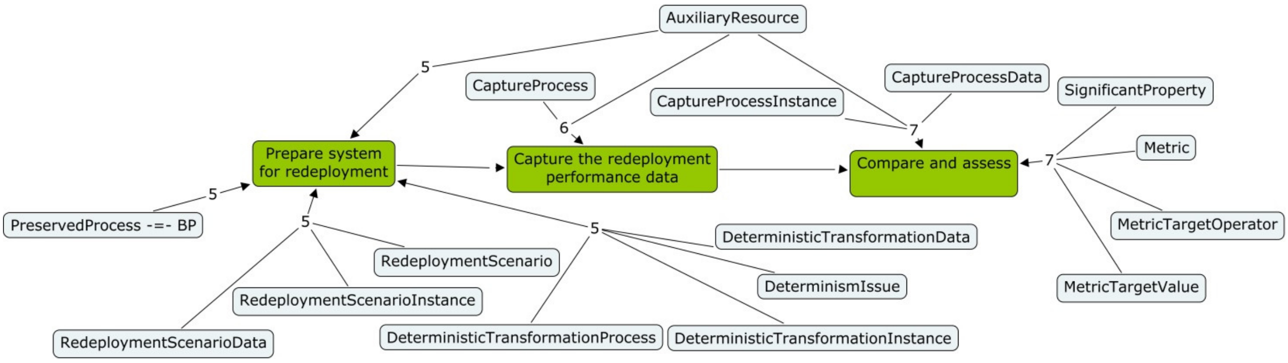


Figure 11: Mapping of the VFramework steps executed in the redeployment environment to the VPlan classes.

In the fifth step of the VFramework, which is *Prepare system for redeployment*, the process is redeployed using information from the process context model. The process instances referred by the VPlan are moved to the system in which they are executed.

In the sixth step of the VFramework, which is *Capture the redeployment performance data*, the capture process which was used in the original environment is used to capture the information from the redeployed process. Sometimes repetition of the exact capture process is impossible, but it is up to the preservation expert to make a decision how to design a new capture process which is compatible with principles of the original one, which is provided by the VPlan.

In the seventh step of the VFramework, which is *Compare and asses*, the final assessment of the redeployment is conducted. Information on metrics, their original values and expected values are obtained from the VPlan.

3.5 VHelper

The automation of the VFramework application and the VPlan creation not only accelerates the verification process, but also decreases the likelihood of human error. The VHelper is a proof of concept tool to demonstrate that the VFramework and the VPlan can easily benefit from automation. Figure 12 summarizes key facts about the VHelper.

VHelper

- automates VPlan creation
- facilitates VFramework application
- uses VPlan to create folder structure
- generates parts of the VPlan
- automatically adds data to the VPlan
- validates data

Figure 12: VHelper key features.

3.5.1 Concept overview

The use case diagram presented in Figure 13 illustrates the application of the VHelper.

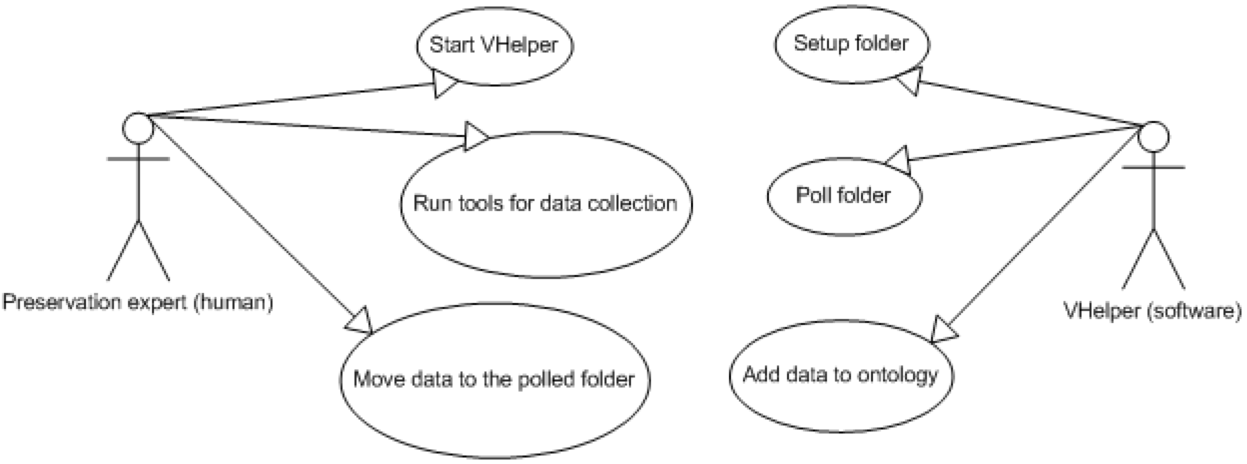

Start VHelper
Setup folder
Run tools for data collection
Poll folder
Preservation expert (human)
VHelper (software)
Move data to the polled folder
Add data to ontology

Figure 13: VHelper use case diagram.

The figure depicts two actors: preservation expert and the VHelper software. The preservation expert runs the VHelper software on the machine where he works on the VPlan. The preservation expert performs steps of the VFramework. When they reach the stage when, for example, the capture processes are defined, then they run the VHelper providing the VPlan as the input. The VHelper creates the folder structure based on defined individuals in the VPlan. Then it starts polling of these folders. The preservation expert extracts the data from the process manually or with a use of any other tools and copies the data into the appropriate folder. The VPlan is automatically updated when new data is detected. The process is repeated until the preservation expert has no data to add. The result of the VHelper execution is the modified VPlan file which holds information on data location, as well as the folder structure with validated data. This process is also depicted in the Figure 14 which shows the activity diagram of the VHelper.

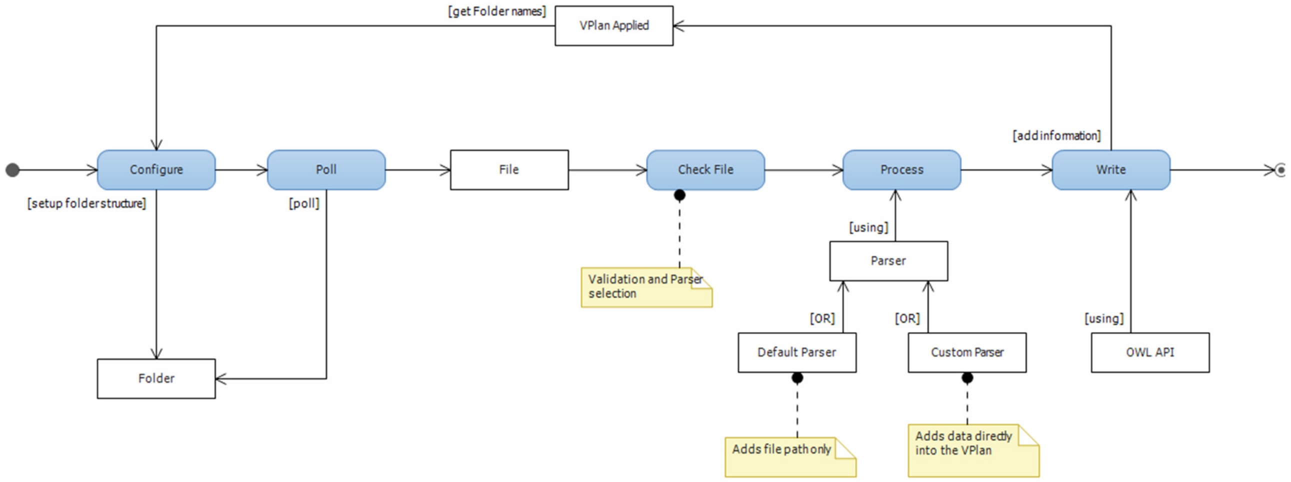


Figure 14: VHelper activity diagram.

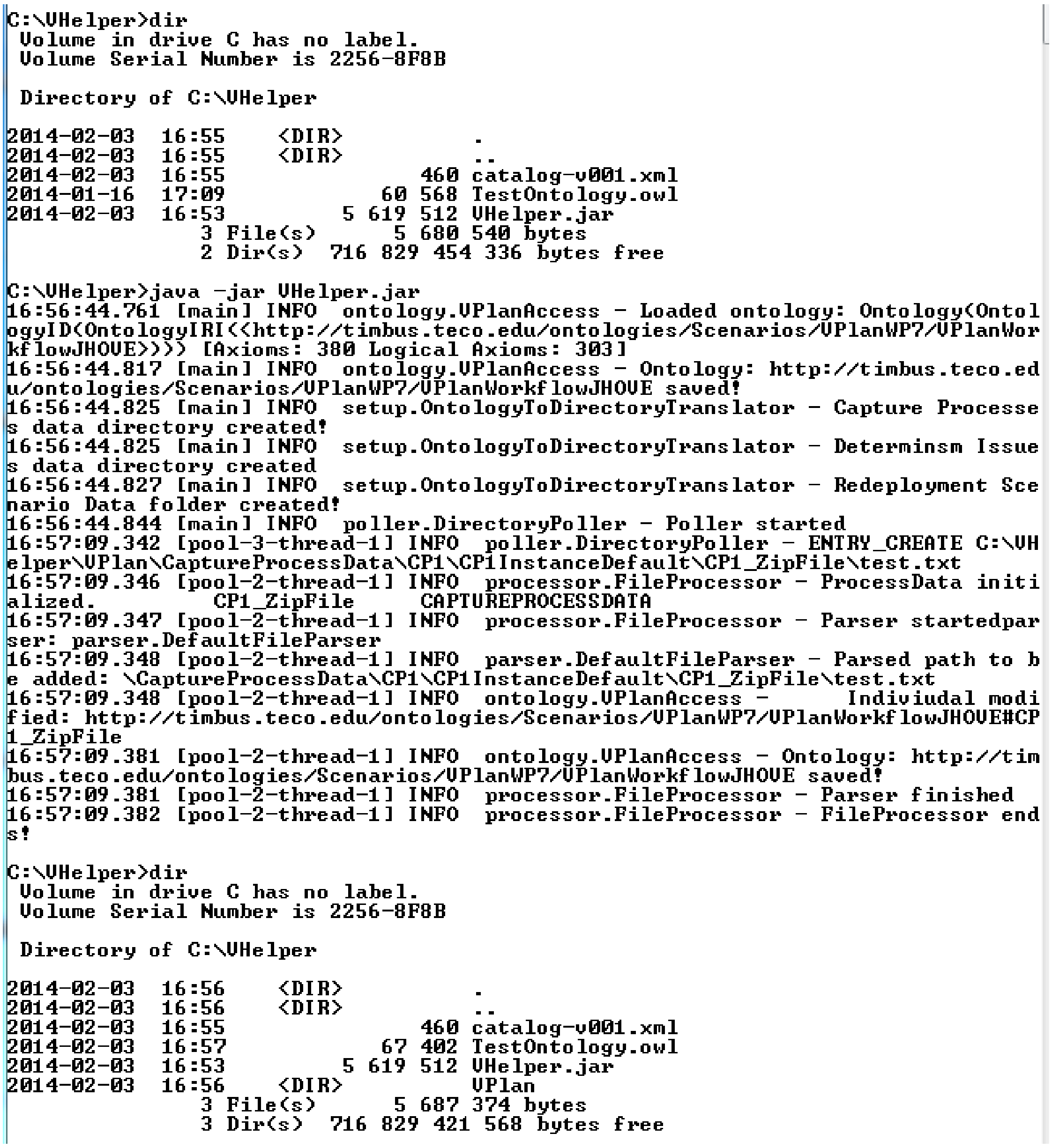
The VHelper can also be customized and therefore provide more functionality in case of custom files. By default, the VHelper uses default validation method and default parsers. This means that for the validation it is checked if the file exists and if it is not empty, while for the parsing part only the path of the file is added to the ontology. In case of a custom validator and parser, it is possible that the VHelper recognizes, for example, the extension of the file and on that basis chooses the appropriate custom validator and parser. Such a custom validator can analyse the file structure and format, while the parser can pre-process the file and store only necessary information from the file directly in the VPlan. There is a wide range of possible implementations and extensions to the proposed architecture of the VHelper.

3.5.2 Implementation

The VHelper proof of concept was implemented with a use of Java 7 and uses OWL API to manipulate the ontology files. It is a console application. The tool is capable of:

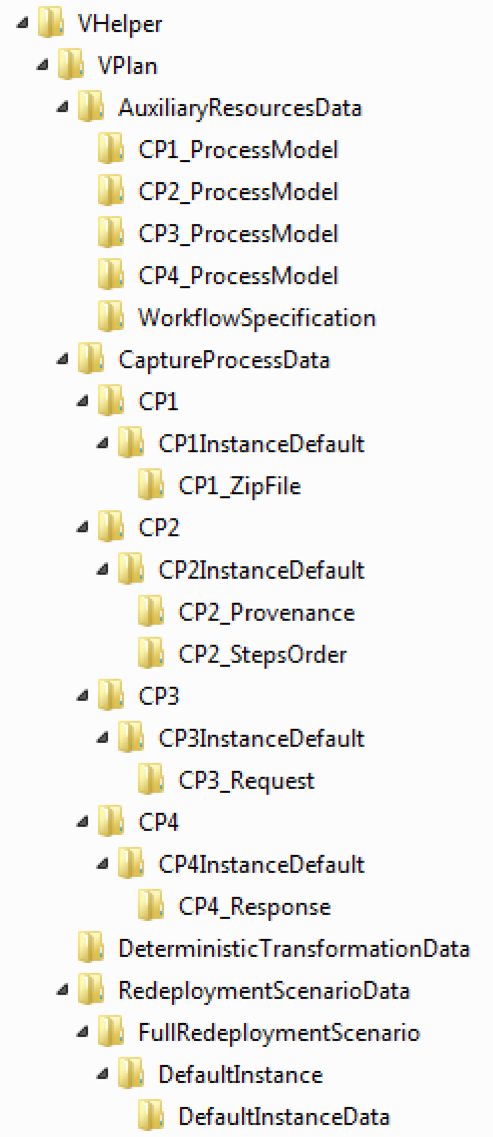
- folder structure creation based on the VPlan,
- multi thread polling of files in the monitored directory,
- validation of pulled data,
- automatic creation of properties and individuals (only for Capture Process),
- adding location of data (only for Capture Process).

Figure 15 presents an example of VHelper execution. The tool was run from the command line. Firstly it created the folder structure (see Figure 16) based on the input ontology file 'TestOntology.owl' (see Figure 17). Then the 'test.txt' file was copied manually to one of the folder. VHelper has instantly detected, validated and added the file to the VPlan. Figure 18 and Figure 19 show the concept maping to the VPlan ontology respectively before and after the changes.



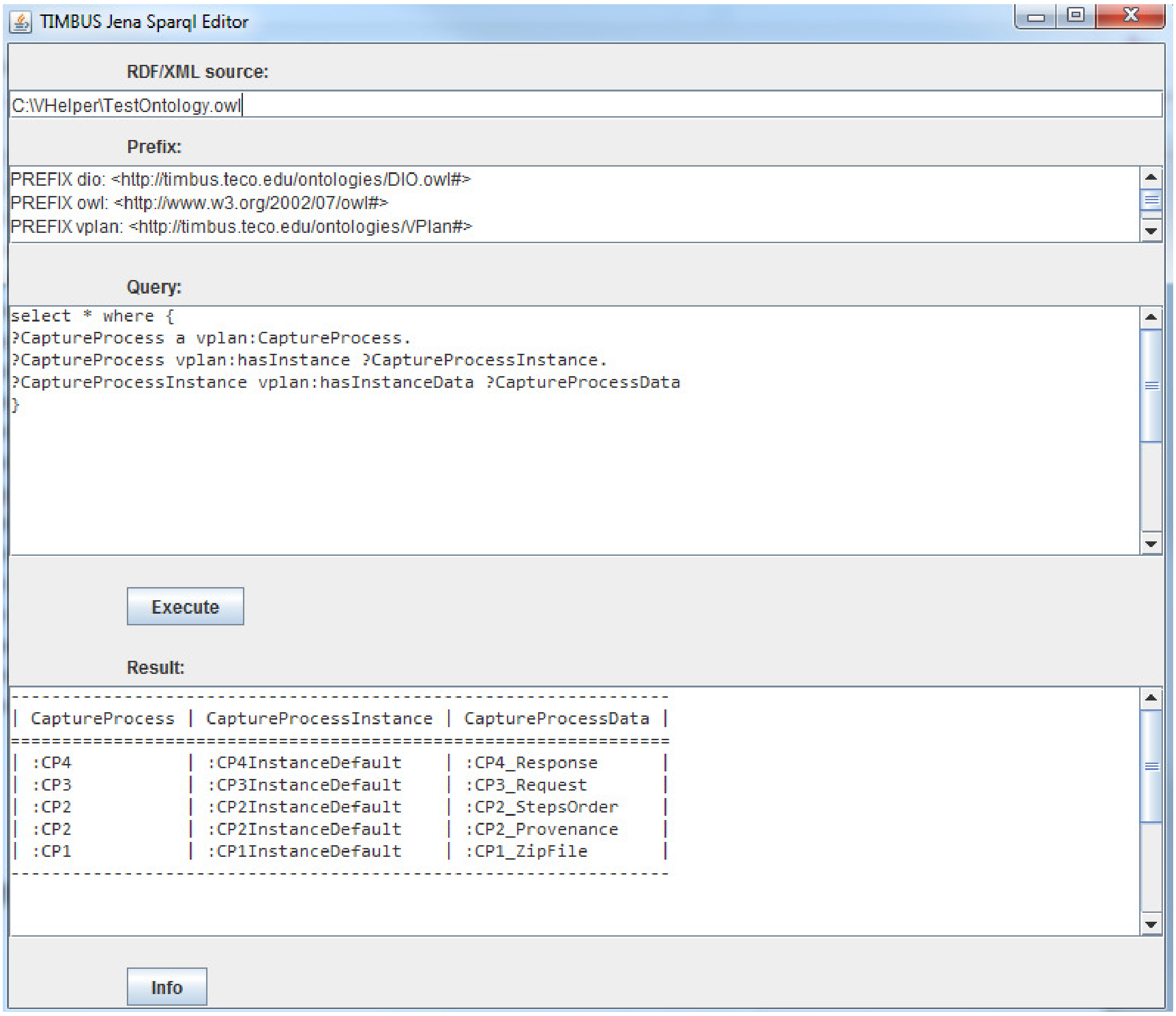
```
C:\VHelper>dir
 Volume in drive C has no label.
 Volume Serial Number is 2256-8F8B

 Directory of C:\VHelper

2014-02-03  16:55    <DIR>          .
2014-02-03  16:55    <DIR>          ..
2014-02-03  16:55               460 catalog-v001.xml
2014-01-16  17:09            60 568 TestOntology.owl
2014-02-03  16:53         5 619 512 VHelper.jar
               3 File(s)      5 680 540 bytes
               2 Dir(s)  716 829 454 336 bytes free

C:\VHelper>java -jar VHelper.jar
16:56:44.761 [main] INFO  ontology.VPlanAccess - Loaded ontology: Ontology(Ontol
ogyID(OntologyIRI(<http://timbus.teco.edu/ontologies/Scenarios/VPlanWP7/VPlanWor
kflowJHOVE>)) [Axioms: 380 Logical Axioms: 303]
16:56:44.817 [main] INFO  ontology.VPlanAccess - Ontology: http://timbus.teco.ed
u/ontologies/Scenarios/VPlanWP7/VPlanWorkflowJHOVE saved!
16:56:44.825 [main] INFO  setup.OntologyToDirectoryTranslator - Capture Processe
s data directory created!
16:56:44.825 [main] INFO  setup.OntologyToDirectoryTranslator - Determinsm Issue
s data directory created
16:56:44.827 [main] INFO  setup.OntologyToDirectoryTranslator - Redeployment Sce
nario Data folder created!
16:56:44.844 [main] INFO  poller.DirectoryPoller - Poller started
16:57:09.342 [pool-3-thread-1] INFO  poller.DirectoryPoller - ENTRY_CREATE C:\VH
elper\VPlan\CaptureProcessData\CP1\CP1InstanceDefault\CP1_ZipFile\test.txt
16:57:09.346 [pool-2-thread-1] INFO  processor.FileProcessor - ProcessData initi
alized.         CP1_ZipFile     CAPTUREPROCESSDATA
16:57:09.347 [pool-2-thread-1] INFO  processor.FileProcessor - Parser startedpar
ser: parser.DefaultFileParser
16:57:09.348 [pool-2-thread-1] INFO  parser.DefaultFileParser - Parsed path to b
e added: \CaptureProcessData\CP1\CP1InstanceDefault\CP1_ZipFile\test.txt
16:57:09.348 [pool-2-thread-1] INFO  ontology.VPlanAccess -      Indiviudal modi
fied: http://timbus.teco.edu/ontologies/Scenarios/VPlanWP7/VPlanWorkflowJHOVE#CP
1_ZipFile
16:57:09.381 [pool-2-thread-1] INFO  ontology.VPlanAccess - Ontology: http://tim
bus.teco.edu/ontologies/Scenarios/VPlanWP7/VPlanWorkflowJHOVE saved!
16:57:09.381 [pool-2-thread-1] INFO  processor.FileProcessor - Parser finished
16:57:09.382 [pool-2-thread-1] INFO  processor.FileProcessor - FileProcessor end
s!

C:\VHelper>dir
 Volume in drive C has no label.
 Volume Serial Number is 2256-8F8B

 Directory of C:\VHelper

2014-02-03  16:56    <DIR>          .
2014-02-03  16:56    <DIR>          ..
2014-02-03  16:55               460 catalog-v001.xml
2014-02-03  16:57            67 402 TestOntology.owl
2014-02-03  16:53         5 619 512 VHelper.jar
2014-02-03  16:56    <DIR>          VPlan
               3 File(s)      5 687 374 bytes
               3 Dir(s)  716 829 421 568 bytes free
```

Figure 15: Example run of VHelper. The folder for storing data was created and one file has been copied to the folder. The file was automatically added to the ontology.



```
VHelper
  VPlan
    AuxiliaryResourcesData
      CP1_ProcessModel
      CP2_ProcessModel
      CP3_ProcessModel
      CP4_ProcessModel
      WorkflowSpecification
    CaptureProcessData
      CP1
        CP1InstanceDefault
          CP1_ZipFile
      CP2
        CP2InstanceDefault
          CP2_Provenance
          CP2_StepsOrder
      CP3
        CP3InstanceDefault
          CP3_Request
      CP4
        CP4InstanceDefault
          CP4_Response
    DeterministicTransformationData
    RedeploymentScenarioData
      FullRedeploymentScenario
        DefaultInstance
          DefaultInstanceData
```

Figure 16: Example of a folder structure created by the VHelper.



RDF/XML source:

C:\VHelper\TestOntology.owl

Prefix:

```
PREFIX dio: <http://timbus.teco.edu/ontologies/DIO.owl#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX vplan: <http://timbus.teco.edu/ontologies/VPlan#>
```

Query:

```
select * where {
?CaptureProcess a vplan:CaptureProcess.
?CaptureProcess vplan:hasInstance ?CaptureProcessInstance.
?CaptureProcessInstance vplan:hasInstanceData ?CaptureProcessData
}
```

Result:

```
-----------------------------------------------------------------
| CaptureProcess | CaptureProcessInstance | CaptureProcessData |
=================================================================
| :CP4           | :CP4InstanceDefault    | :CP4_Response      |
| :CP3           | :CP3InstanceDefault    | :CP3_Request       |
| :CP2           | :CP2InstanceDefault    | :CP2_StepsOrder    |
| :CP2           | :CP2InstanceDefault    | :CP2_Provenance    |
| :CP1           | :CP1InstanceDefault    | :CP1_ZipFile       |
-----------------------------------------------------------------
```

Figure 17: SPARQL query and its results. The result presents Capture Processes, Capture Process Instances and Capture Process Data. They have been used to create the folder structure presented in Figure 16.

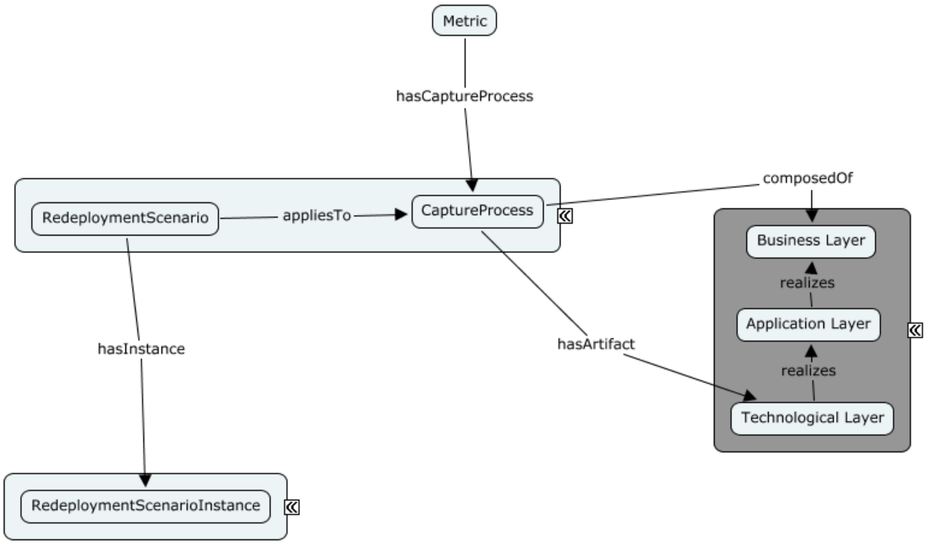


Figure 18: Concept map depicting the VPlan ontology showing only concepts relevant for automatic addition of information to Capture Process.

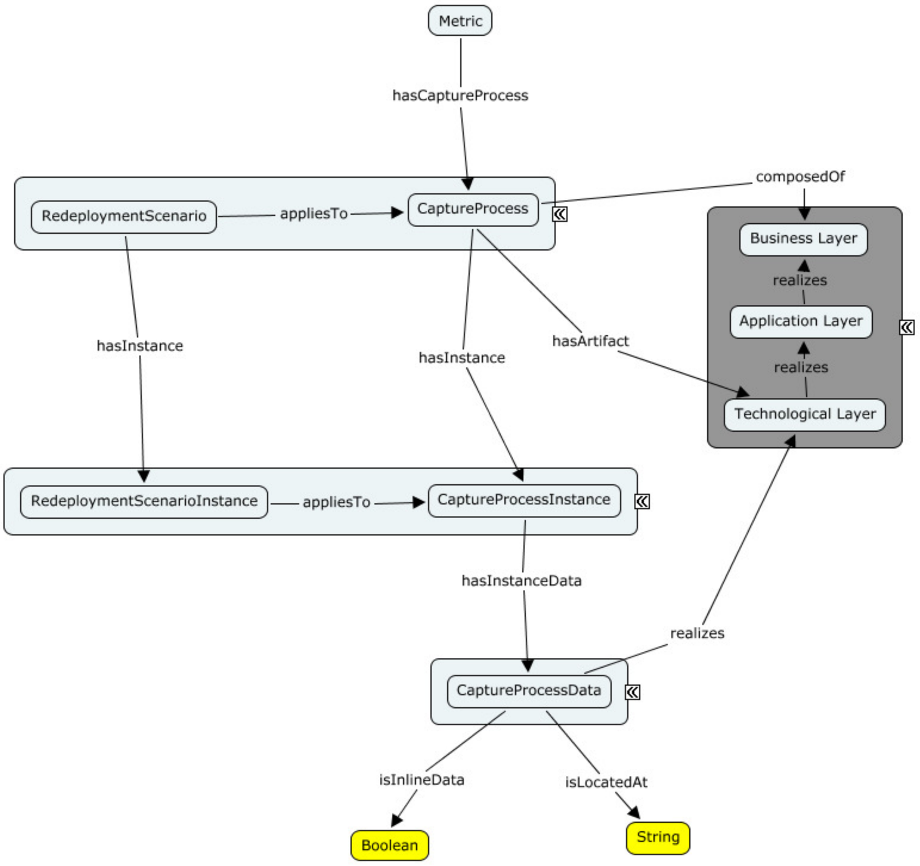


Figure 19: Concept map depicting the VPlan ontology after automatic addition of information by VHelper to Capture Process.

3.6 SPARQL queries

SPARQL[7] is a query language that allows retrieving RDF[8] data. The VPlan and the TIMBUS context model are RDF compatible and for this reason the SPARQL queries can be used in the process of verification and validation. This section presents sample queries and highlights the benefits they bring on various stages of the VFramework application. More examples of queries can be found in Annex A.

3.6.1 Validation of the VPlan

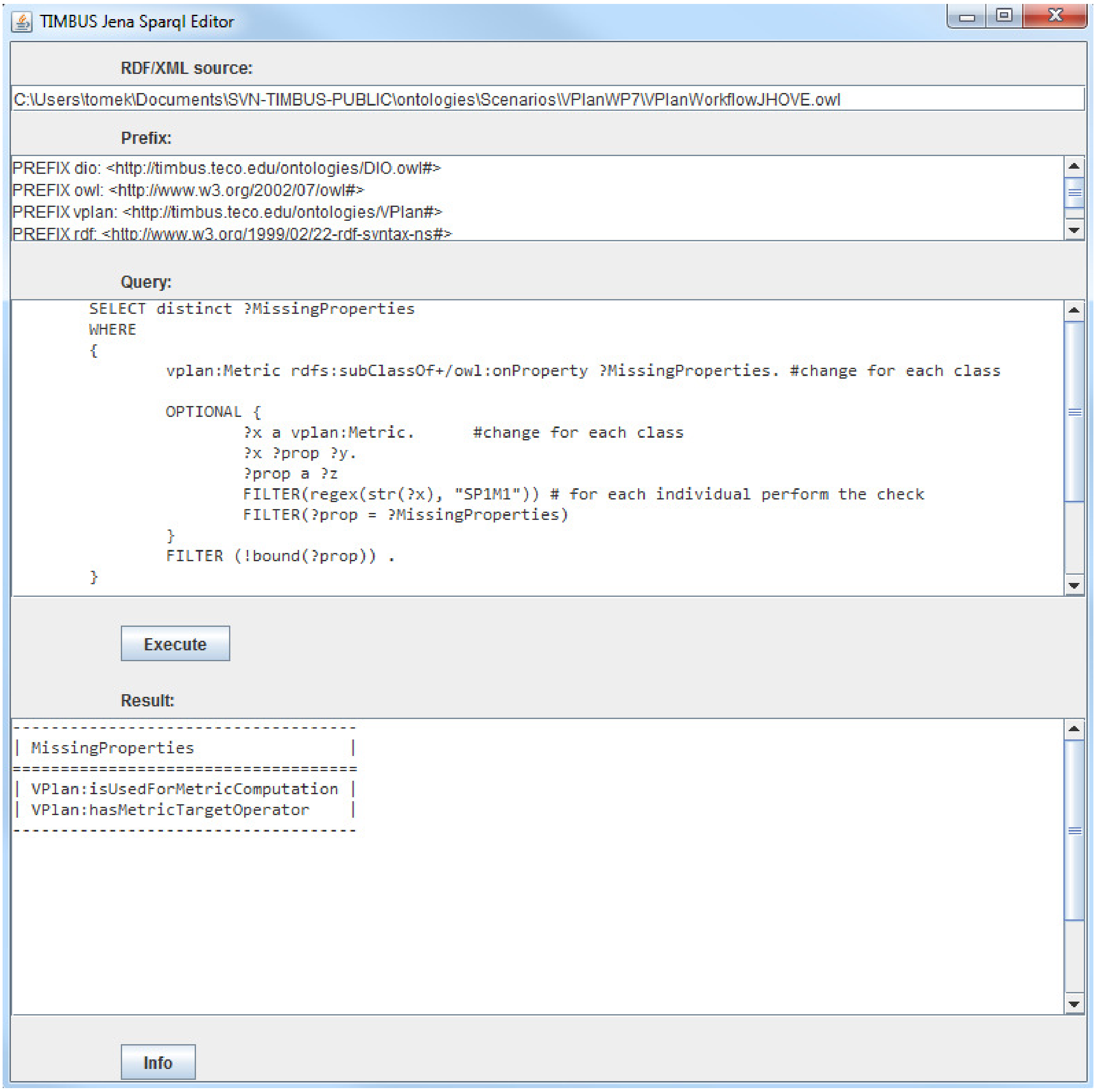
When the data is collected in the original environment (step 4 of the VFramework) it is essential to validate it. This requires manual or automatic actions and varies depending on the data type. The VHelper (see Section 3.5) is a tool which can perform such checks. However, the validation of captured data is not sufficient. In the proposed solution the VPlan plays a central role of a deposit in which all the information about the preserved business process is stored. Furthermore, in case when only parts of the VPlan are automatically generated and the rest is created manually there is a risk of errors in the model introduced by a human. Therefore, the validation of the VPlan is also needed.

The SPARQL queries can be used for validation of the VPlan. The combination of queries is capable of checking the model completeness, i.e. detecting if the object or data properties of the VPlan instance comply with the VPlan specification, or if the object properties between the individuals of the VPlan instance are allowed. In Figure 20 an example of validation query is presented.

The query validates the instance of the VPlan. In this particular case the individual of a class Metric which name contains “SP1M1” is validated, i.e. it is checked if this individual has all its properties specified according to the VPlan specification of the class Metric. The example execution of the query is presented in the Figure 20. One can notice, that the result revealed that two properties are missing, namely: *VPlan:isUsedForMetricComputation* and *VPlan:hasMetricTargetOperator*. Hence, the VPlan is not valid and this issue needs to be fixed.

[7] SPARQL: http://www.w3.org/TR/rdf-sparql-query/

[8] RDF: http://www.w3.org/RDF/



C:\Users\tomek\Documents\SVN-TIMBUS-PUBLIC\ontologies\Scenarios\VPlanWP7\VPlanWorkflowJHOVE.owl

```
PREFIX dio: <http://timbus.teco.edu/ontologies/DIO.owl#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX vplan: <http://timbus.teco.edu/ontologies/VPlan#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
```

```
SELECT distinct ?MissingProperties
WHERE
{
        vplan:Metric rdfs:subClassOf+/owl:onProperty ?MissingProperties. #change for each class

        OPTIONAL {
                ?x a vplan:Metric.      #change for each class
                ?x ?prop ?y.
                ?prop a ?z
                FILTER(regex(str(?x), "SP1M1")) # for each individual perform the check
                FILTER(?prop = ?MissingProperties)
        }
        FILTER (!bound(?prop)) .
}
```

```
------------------------------------
| MissingProperties                |
====================================
| VPlan:isUsedForMetricComputation |
| VPlan:hasMetricTargetOperator    |
------------------------------------
```

Figure 20: Example of SPARQL validation query execution. The result reveals that at least two properties are not defined in the VPlan instance.

In the provided example, we have focused on a single class and a single individual. Such a validation is possible for all the classes and individuals of the VPlan instance. Such queries can be executed manually using software like Protégé, but preferably their execution should also be automated. There are many programming frameworks which allow this, e.g. combination of Java and Apache Jena [9]. Moreover, not only the execution of the queries can be automated but also their generation. The above presented query can be transformed into a template and queries for validation of further classes and

[9] Apache Jena: http://jena.apache.org/

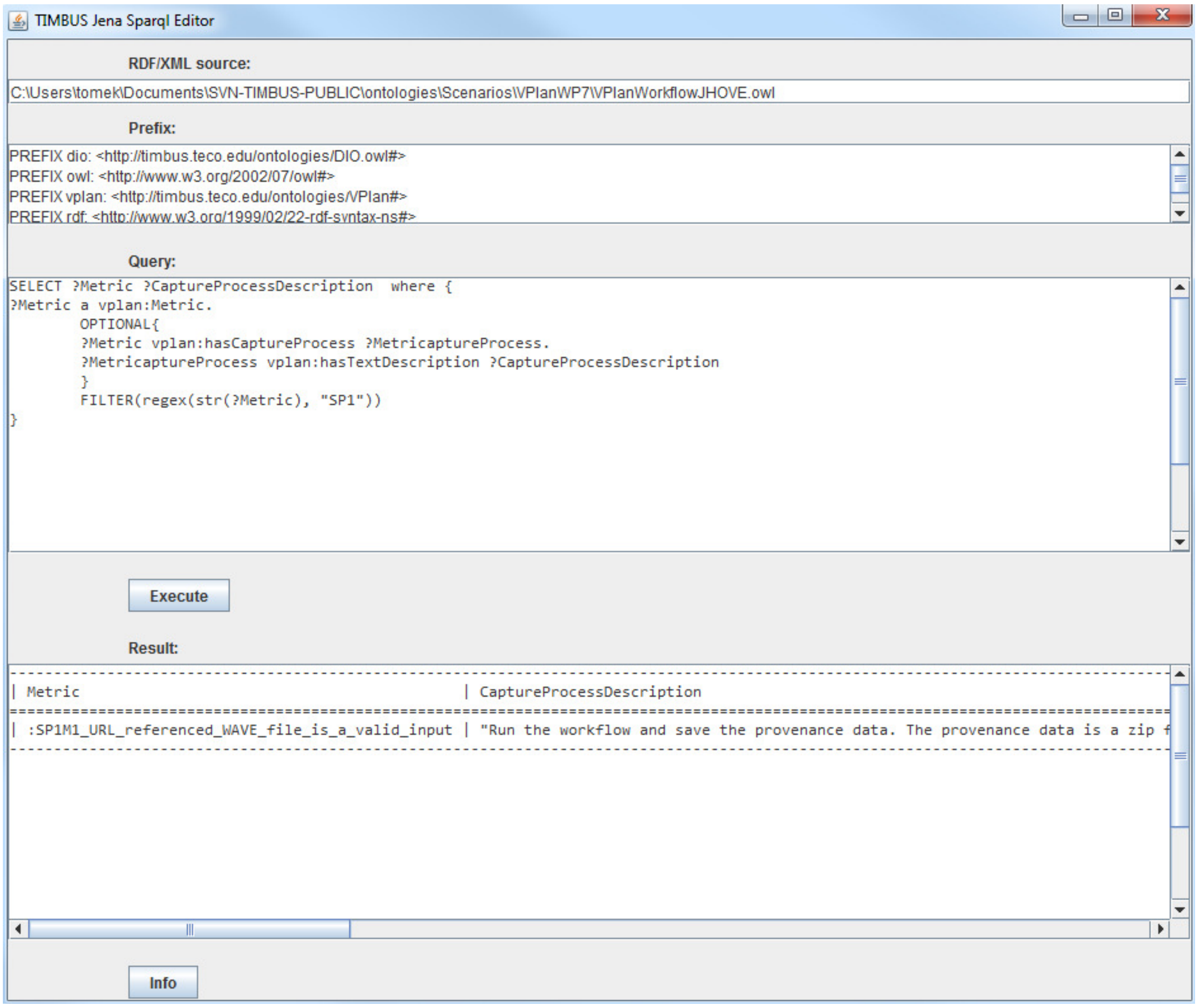
individuals can be generated. However, such a tool is not in the scope of this task and therefore the presented query has to be considered as a proof of concept.

3.6.2 Reporting

SPARQL queries can also facilitate presentation of data stored in the VPlan. During the redeployment phase of the VFramework the preservation expert must make many decisions concerning the redeployment using the information about the original process. For example, the capturing processes in the redeployment environment have to mimic the capturing processes from the original environment. Furthermore, when in the last step of the VFramework the assessment of the redeployment is performed, the information on: metrics, target values and expected values, in the redeployment are needed. In both cases this information was defined during the first phase of the VFramework and is stored within the VPlan. Hence, at the different stages of the VFramework application needs to obtain different information from the VPlan. Using of SPARQL queries is a convenient way of doing this. .

Figure 21 presents an example of a SPARQL query which was executed in order to get the description of the capture process for a given metric. Such information is needed at the “Prepare system for redeployment” step of the VFramework. The given example is only a small sample of what is needed during the verification of the process. In general the SPARQL queries executed on the VPlan can provide much more information, e.g. they can provide answers to the following questions (and are not limited to them only).

- What are the significant properties of the process?
- What are the metrics used to measure the significant properties?
- What is the capture process for each of the metrics?
- Where are the auxiliary resources fostering understanding of the capture process located?
- Where are the measurement points in the process?
- Which data has been collected for calculating metrics?
- What are the expected values of the metrics?
- Where is the captured data?
- What instances of the process are used for verification?
- Are there any determinism issues in the process?
- What are the process steps?
- Are there any auxiliary descriptions of the process?



RDF/XML source:

C:\Users\tomek\Documents\SVN-TIMBUS-PUBLIC\ontologies\Scenarios\VPlanWP7\VPlanWorkflowJHOVE.owl

Prefix:

```
PREFIX dio: <http://timbus.teco.edu/ontologies/DIO.owl#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX vplan: <http://timbus.teco.edu/ontologies/VPlan#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
```

Query:

```
SELECT ?Metric ?CaptureProcessDescription  where {
?Metric a vplan:Metric.
        OPTIONAL{
        ?Metric vplan:hasCaptureProcess ?MetricaptureProcess.
        ?MetricaptureProcess vplan:hasTextDescription ?CaptureProcessDescription
        }
        FILTER(regex(str(?Metric), "SP1"))
}
```

Result:

```
------------------------------------------------------------------------------------------------------------------------
| Metric                                          | CaptureProcessDescription
========================================================================================================================
| :SP1M1_URL_referenced_WAVE_file_is_a_valid_input | "Run the workflow and save the provenance data. The provenance data is a zip f
------------------------------------------------------------------------------------------------------------------------
```

Figure 21: Example of SPARQL query used for presentation of data stored in a VPlan instance and its result. The result presents that significant property SP6_Result_file_correctness has 4 metrics measuring its fulfilment.

3.7 Use case: application on LNEC2

This section presents the application of the developed concepts on a specific process from the civil engineering use case addressed in WP8. We isolated a specific process to better narrow the verification solution proposed in TIMBUS. This allows us to limit and control the process executions and verify the produced results.

The following sub sections detail the process and detail the proposed verification applied. We present the application of the VFramework in both environments: the original and the redeployment environment. The redeployment environment is simulated by redeploying the process in a substantially different environment than the original one.

3.7.1 Use case description

The use case addressed in this section is related to the overall civil engineering processes described in deliverables D8.1: Use case definition and digital preservation requirements and D8.4 Digital preservation of CAD/CAM business processes (TIMBUS Consortium, 2012).

The safety control of large dams is based on the monitoring of important physical quantities that characterize the structural behaviour. The analysis of data captured by the monitoring systems and their comparison with statistical, physical and mathematical models is critical for the safety control assessment. It is known that the variations of hydrostatic pressure and temperature are the main actions that must be considered when analysing the physical quantities generated by the monitoring systems. As a consequence, multiple linear regressions (MLR) can be used to determine their relationship with the expected response (physical quantity).

In large dams, the expected response is approximated by the following effects: (*i*) elastic effect of the hydrostatic pressures; (*ii*) elastic effect of temperature, depending on thermal conditions; and (*iii*) time effect (considered irreversible).

Figure 22 details a multiple linear regression process used in dam safety. For demonstration purposes, this process was isolated from the generic information system (*GestBarragens*[10]). Overall, the process is composed by 5 steps:

- Extract data: based on a set of extraction parameters, this process generates the sensor data that will be used in the MLR model;
- Generate regression: Based on a set of regression parameters (e.g., equation to estimate the hydrostatic effect), this process generates the regression controls that configure the parameters for the MLR model;
- Execute regression: This process executes the regression parameterized in the regression control, using the dataset generated in the extract data process;

[10] *GestBarragens* is detailed in deliverable D8.1 – Use case definition and digital preservation requirements

- Generate aggregation: since a dam has a large number of sensors and a regression is used for each physical quantity associated with each sensor, we might need to run hundreds or thousands of regressions. Thus, the process is capable to aggregate all MLR executions into one aggregated report. This step generates the controls that define how this data is aggregated;
- Produce report: collects all the results produced by the several executions of MLR models and compile them into a single report.

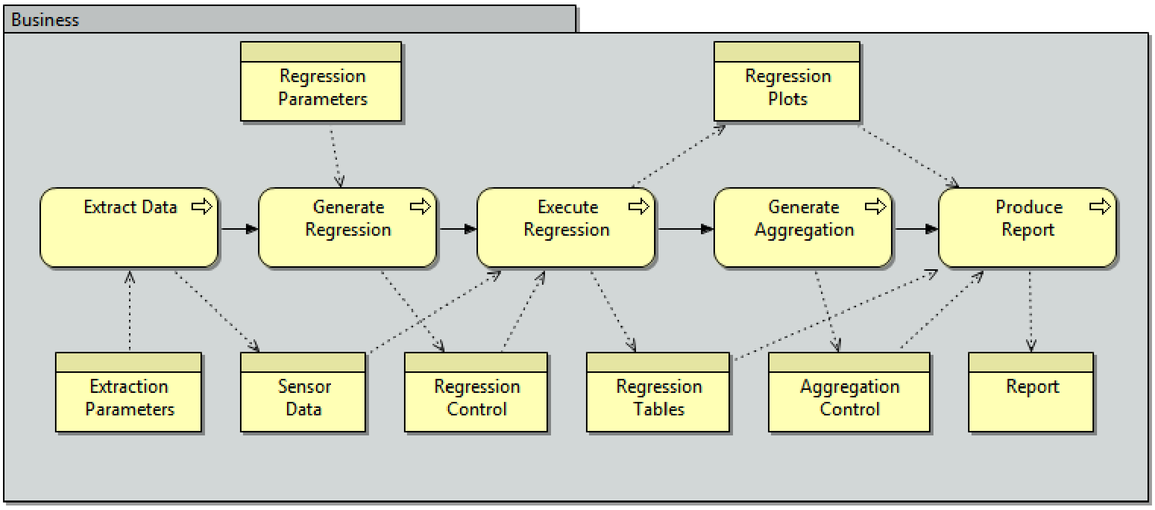


Figure 22 – Multiple linear regression process in dam safety.

3.7.2 VFramework application

This section explains the application of the verification process to the original environment by showing how each of the VFramework steps was executed.

3.7.2.1 VFramework step 1 – Describe the original environment

Following the proposal described in Section 3.4.3 we have initialized a clean ontology file in the Protégé ontology editor. Next, the process must be described in order to populate the ontology. In this case, the process was manually modelled using the Archi tool. Figure 23 depicts the context model of the MLR process, detailing the business, application and technology layers. Finally, the context model was converted[11] and imported into the VPlan.

[11] Archi2Owl converter: https://opensourceprojects.eu/p/timbus/context-model/converters/archi2owl/ci/725f97dc9ad47fa1bae6bdacc83384e12785015e/tree/

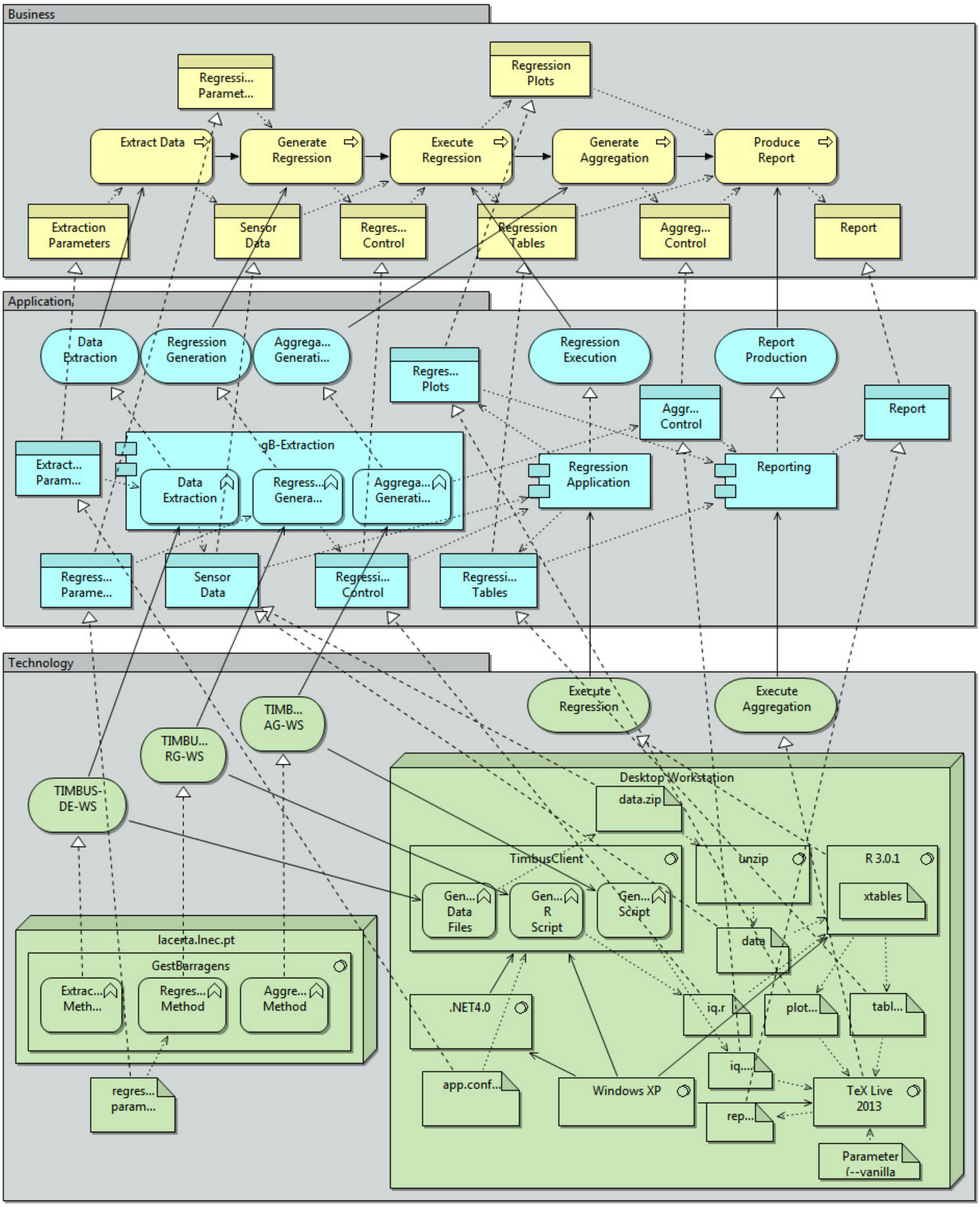


Figure 23: Detailed ArchiMate model of the MLR process before conversion to the ontology.

Then we defined one redeployment scenario for this MLR process, which assumes that the process will be rerun in the future including all its steps. We assume that the scenario will be redeployed in order to reproduce the original behaviour.

For verification we opted to execute only one instance. The instance consists only of the extraction parameters (app.config at the technological level) required to configure and run the steps of this process.

The last action was specification of significant properties which are presented in Table 1. Each of them has its id, name and description.

Table 1: Significant properties defined for the MLR process.

ID: SP1	**Name:** Generate Data
Short description: The system must be able to generate sensor data for quantitative interpretation	
Additional information: Is the generated data complete and correct?	
ID: SP2	**Name:** Export by
Short description: The system must export the data by, at least: – Structure – Date Period – Sensor Type	
Additional information: Can we export data by the specified parameters?	
ID: SP3	**Name:** Quantitative Interpretation
Short description: The system must be able to execute the quantitative interpretation for all the physical quantities of the selected sensor type	
Additional information: Are all the physical quantities quantified?	
ID: SP4	**Name:** Coefficients
Short description: The system must provide the coefficients used in the interpretation, mainly: – Estimate – Standard Error – t value – Pr(>|t|)	
Additional information: Are all the coefficients computed?	
ID: SP5	**Name:** Quality Measures
Short description: The system must provide the quality measures of the regression, mainly: – Standard Deviation – R2 – Adjusted R2	
Additional information: Are all quality measures computed?	
ID: SP6	**Name:** Residuals
Short description: The system must provide the residuals of the regression in a table	
Additional information: Are all the residuals computed?	

ID: SP7	**Name:** ANOVA Matrix
Short description: The system must provide the ANOVA matrix of the regression	
Additional information: Are all the ANOVA matrixes computed?	
ID: SP8	**Name:** Analysis Concepts
Short description: The system must provide graphical representation of the following concepts: – Thermal effect – Water level effect – Thermal effect/residuals relation – Real/estimated observations – Residual vs Fitted values – Normal Q-Q – Scale-Location – Cook's distance – Residuals vs Leverage – Cook's distance vs Leverage	
Additional information: Are all graphical representations provided?	
ID: SP9	**Name:** Report
Short description: The output of the process should be compiled into a single pdf report	
Additional information: Is the output of the system compiled in one pdf file?	

All of the information collected at this step has been added to the VPlan. The state of the VPlan after execution of the first step is depicted in Figure 24. In order to enhance readability the elements of the preserved process which was imported to the VPlan are not depicted.

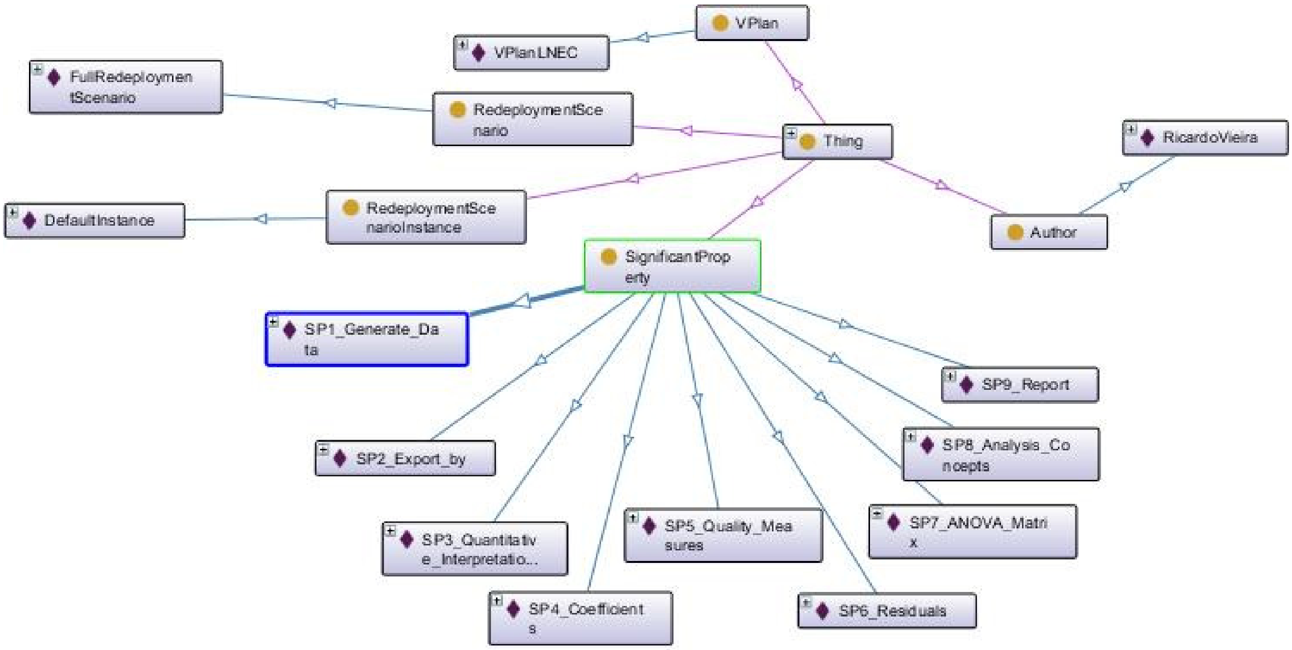


Figure 24: Simplified visualisation of the VPlan after the first step of the VFramework.

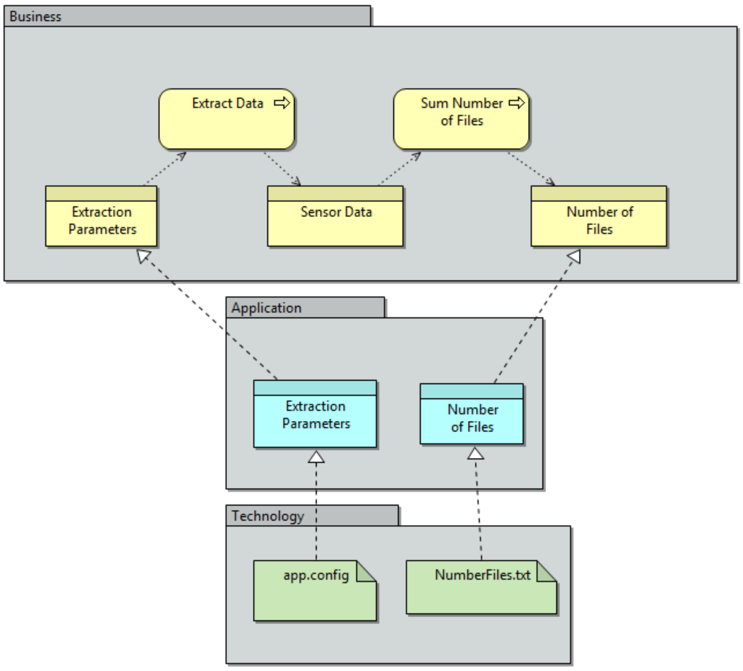
3.7.2.2 VFramework step 2 – Prepare system for preservation

We have analysed the MLR process regarding dependencies or deterministic issues. We concluded that the MLR process is deterministic so no deterministic transformation is necessary. In terms of dependencies, the process has three external dependencies on web-services that are required to execute the process. It is beyond the scope of the verification task to decide whether the web service will be available during the redeployment or if it also has to be preserved. The VFramework states that this web service is needed and therefore is present in the context model. No changes to the VPlan were necessary at this step.

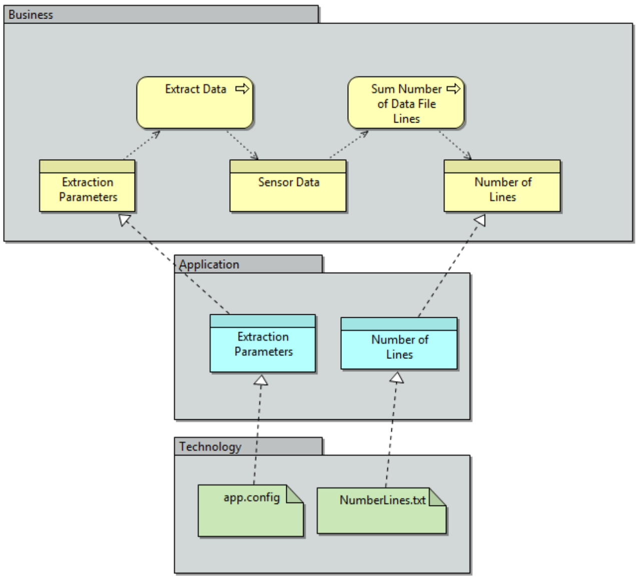
3.7.2.3 VFramework step 3 – Design verification setting

In the third step of the VFramework we have assigned metrics to the significant properties. In some cases more than one metric is used to measure the significant property, while in other cases the same metric can be used to measure two or more significant properties. Furthermore, for each of the metrics we have assigned their target operators and target values which will be used for the assessment during the redeployment. For each of the metrics a capture process was assigned. In some cases the same capture process is used for different metrics. Table 2 provides, for each of the significant properties, the metrics required to evaluate the significant property. Each metric has a name, a description, a target operator, a target value and a capture process required to assess the metric.

Table 2: Significant properties, metrics and capture processes of the MLR process.

ID: SP1	**Name:** Generate Data
Short description: The system must be able to generate sensor data for quantitative interpretation	
Additional information: Is the generated data complete and correct?	
Metric 1 – Number of Data Files	
Target Operator: Equal	
Target Value: NA	
Capture Process:  Business Extract Data Sum Number of Files Extraction Parameters Sensor Data Number of Files Application Extraction Parameters Number of Files Technology app.config NumberFiles.txt **Description**: Extract Data and count the number of data files that were extracted	
Metric 2 – Total Number of Data File Lines	
Target Operator: Equal	
Target Value: NA	

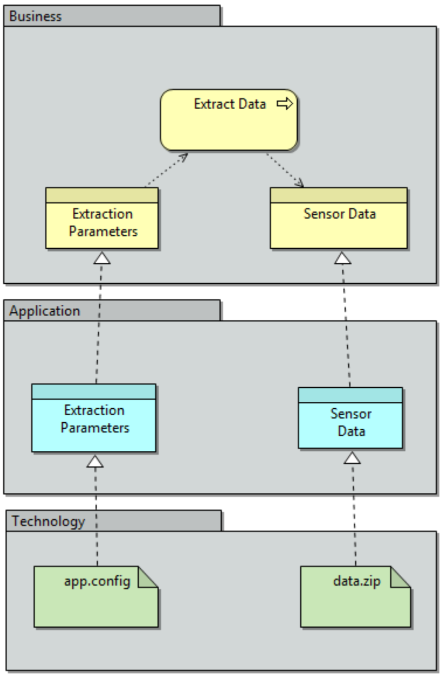
Capture Process:



Description: Extract Data and sum the number of lines in the data files that were extracted

ID: SP2	**Name:** Export by
Short description: The system must export the data by, at least: – Structure – Date Period – Sensor Type	
Additional information: Can we export data by the specified parameters?	
Metric 1 – Numbers of observations not belonging to selected dam	
Description: Verify that all extracted observation of the sensor data correspond to the selected dam in the extraction parameters.	
Target Operator: Exact	
Target Value: 0	

Capture Process:



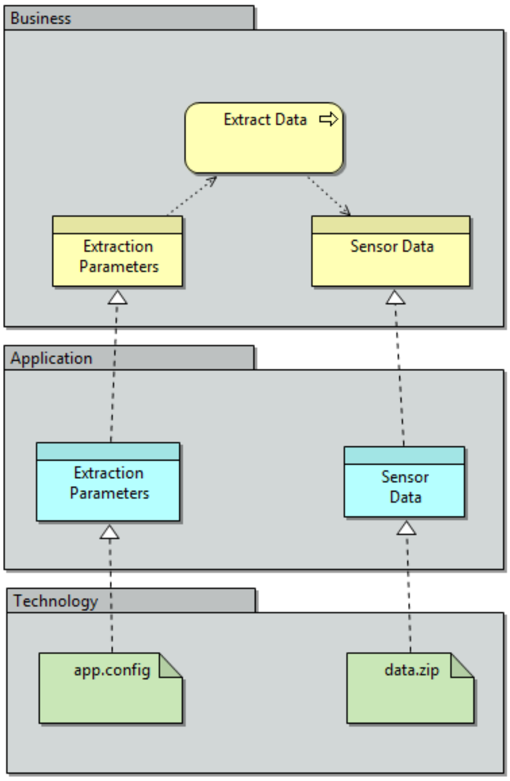
Description: Execute the extract data process with extraction parameters specifying one dam, one date period and one sensor type.

Metric 2 – Numbers of observations not belonging to selected date period

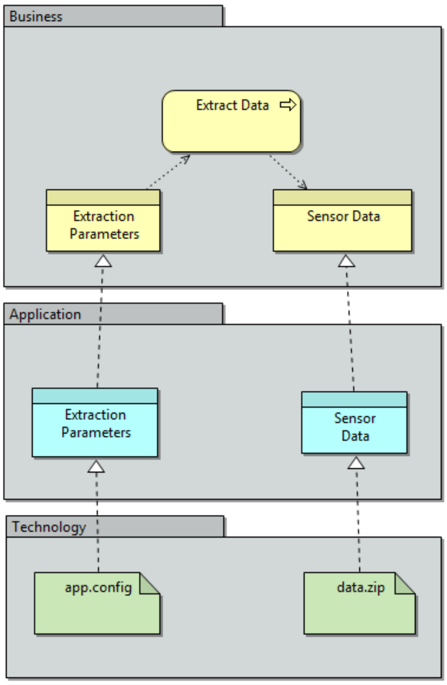
Description: Verify that all extracted observation of the sensor data correspond to the selected date period in the extraction parameters.

Target Operator: Exact

Target Value: 0

Capture Process:  Business Extract Data Extraction Parameters Sensor Data Application Extraction Parameters Sensor Data Technology app.config data.zip **Description**: Execute the extract data process with extraction parameters specifying one dam, one date period and one sensor type.
Metric 3 – Numbers of observations not belonging to selected sensor type
Description: Verify that all extracted observation of the sensor data correspond to the selected sensor type in the extraction parameters.
Target Operator: Exact
Target Value: 0

Capture Process:



Description: Execute the extract data process with extraction parameters specifying one dam, one date period and one sensor type.

ID: SP3	**Name:** Quantitative Interpretation

Short description: The system must be able to execute the quantitative interpretation for all the physical quantities of the selected sensor type

Additional information: Are all the physical quantities quantified?

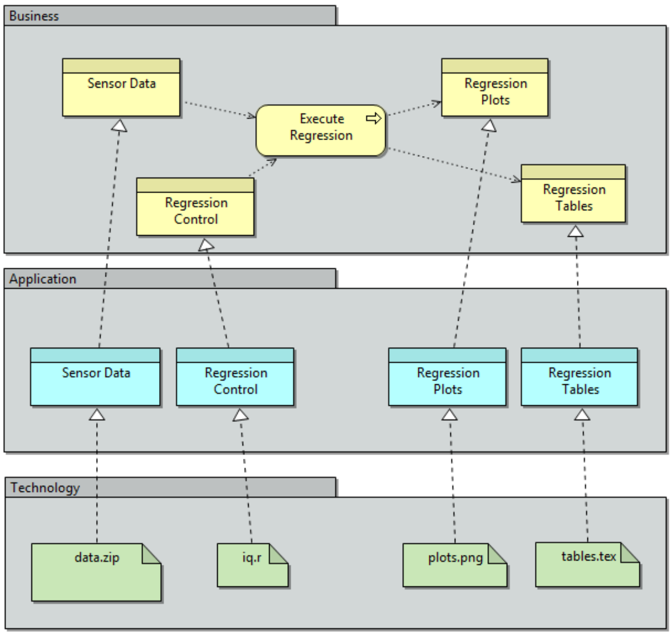
Metric 1 – Numbers of physical quantities not computed

Description: A sensor plot must be generated for each physical quantity present in the sensor data. The number of physical quantities not computed will be the difference between the expected generated regression plots (number of sensor x number of physical quantities) and the generated regression plots.

Target Operator: Exact

Target Value: 0

Capture Process:



Description: Execute the "Execute Regression".

ID: SP4 | **Name:** Coefficients

Short description: The system must provide the coefficients used in the interpretation, mainly:

- Estimate
- Standard Error
- t value
- Pr(>|t|)

Additional information: Are all the coefficients computed?

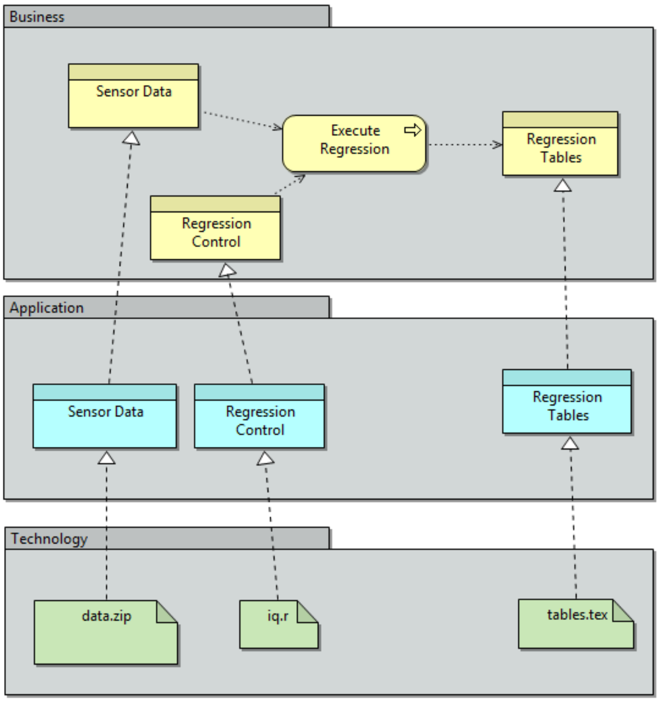
Metric 1 – Numbers of coefficients not computed

Description: The system must provide the coefficients (Estimate, Standard Error, t value, Pr(>|t|) used in the interpretation. The number of coefficients not computed per sensor will be the difference between the expected generated coefficients (number of sensor x number of physical quantities x 4 (number of coefficients)) and the generated coefficients in the regression tables.

Target Operator: Exact

Target Value: 0

Capture Process:



Description: Execute the "Execute Regression".

ID: SP5	**Name:** Quality Measures

Short description: The system must provide the quality measures of the regression, mainly:

- Standard Deviation
- R2
- Adjusted R2

Additional information: Are all quality measures computed?

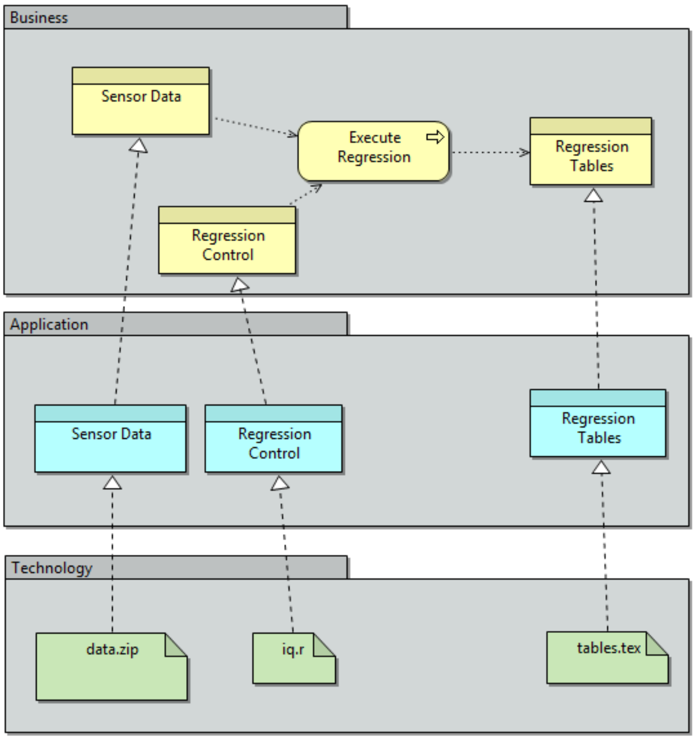
Metric 1 – Numbers of quality measures not computed

Description: The system must provide the quality measures (Standard Deviation, R2, Adjusted R2) used in the interpretation. The number of quality measures not computed per sensor will be the difference between the expected generated quality measures (number of sensor x number of physical quantities x 3 (number of quality measures)) and the generated quality measures in the regression tables.

Target Operator: Exact

Target Value: 0

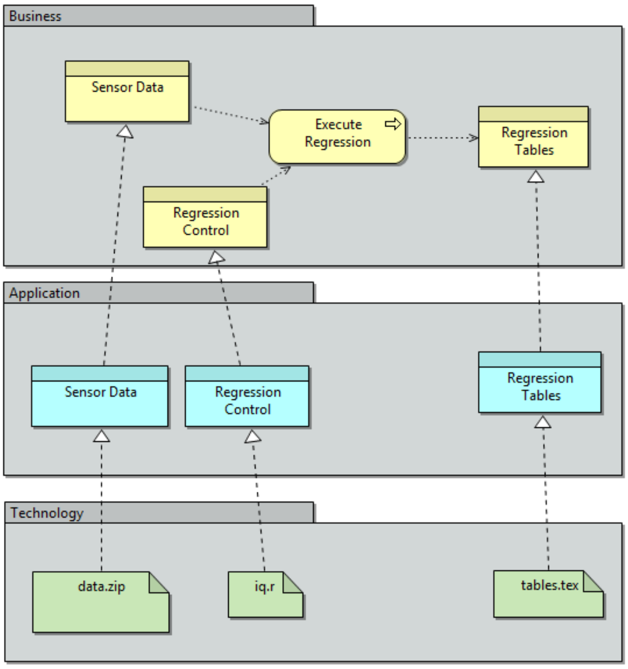
Capture Process:



Description: Execute the “Execute Regression”.

ID: SP6	**Name:** Residuals
Short description: The system must provide the residuals of the regression in a table	
Additional information: Are all the residuals computed?	
Metric 1 – Numbers of residuals not computed	
Description: The system must provide the residuals used in the interpretation. The number of residuals not computed per sensor will be the difference between the expected generated residuals (number of sensor x number of physical quantities) and the generated residuals in the regression tables.	
Target Operator: Exact	
Target Value: 0	

Capture Process:



Description: Execute the “Execute Regression”.

ID: SP7	**Name:** ANOVA Matrix

Short description: The system must provide the ANOVA matrix of the regression

Additional information: Are all the ANOVA matrixes computed?

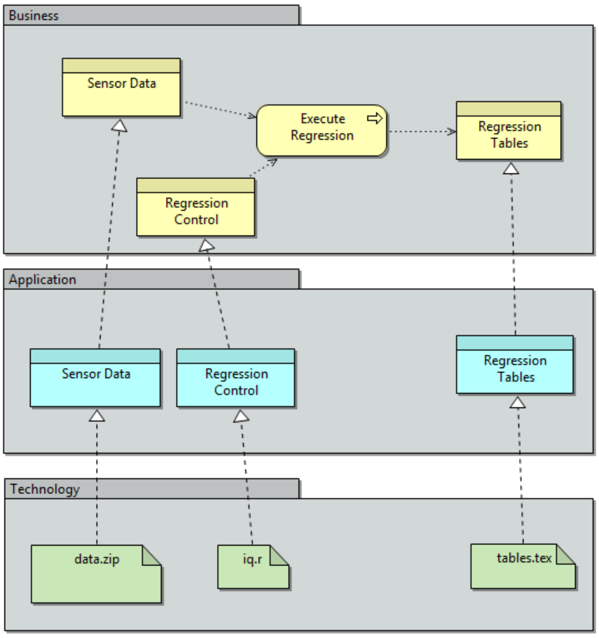
Metric 1 – Numbers of ANOVA Matrix not computed

Description: The system must provide the ANOVA matrixes used in the interpretation. The number of ANOVA matrixes not computed per sensor will be the difference between the expected generated ANOVA matrixes (number of sensor x number of physical quantities) and the generated ANOVA matrixes in the regression tables.

Target Operator: Exact

Target Value: 0

Capture Process:



Description: Execute the “Execute Regression”.

ID: SP8 | **Name:** Analysis Concepts

Short description: The system must provide graphical representation of the following concepts:

- Thermal effect
- Water level effect
- Thermal effect/residuals relation
- Real/estimated observations
- Residual vs Fitted values
- Normal Q-Q
- Scale-Location
- Cook's distance
- Residuals vs Leverage
- Cook's distance vs Leverage

Additional information: Are all graphical representations provided?

Metric 1 – Numbers of analysis concepts not computed

Description: The system must provide graphical representations of the following analysis concepts:

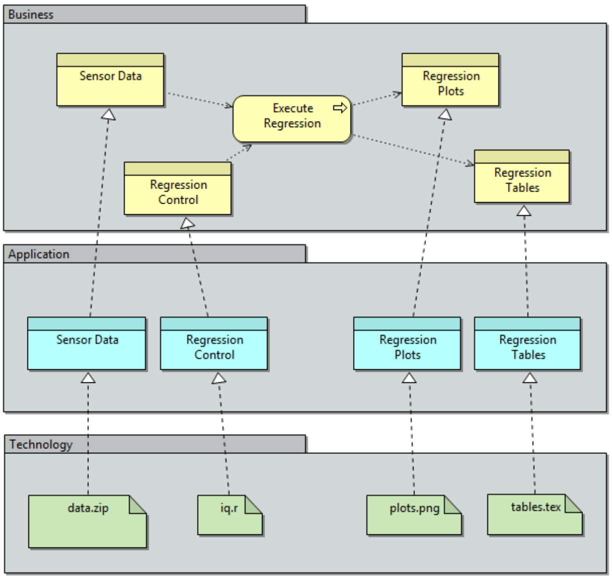
- Thermal effect
- Water level effect
- Thermal effect/residuals relation
- Real/estimated observations
- Residual vs Fitted values
- Normal Q-Q
- Scale-Location
- Cook's distance
- Residuals vs Leverage
- Cook's distance vs Leverage.

The number of analysis concepts not computed per sensor will be the difference between the expected generated analysis concepts (number of sensor x number of physical quantities x 10 (number of analysis concepts)) and the generated analysis concepts in the regression plots.

Target Operator: Exact

Target Value: 0

Capture Process:



Description: Execute the “Execute Regression”.

ID: SP9	**Name:** Report

Short description: The output of the process should be compiled into a single pdf report

Additional information: Is the output of the system compiled in one pdf file?

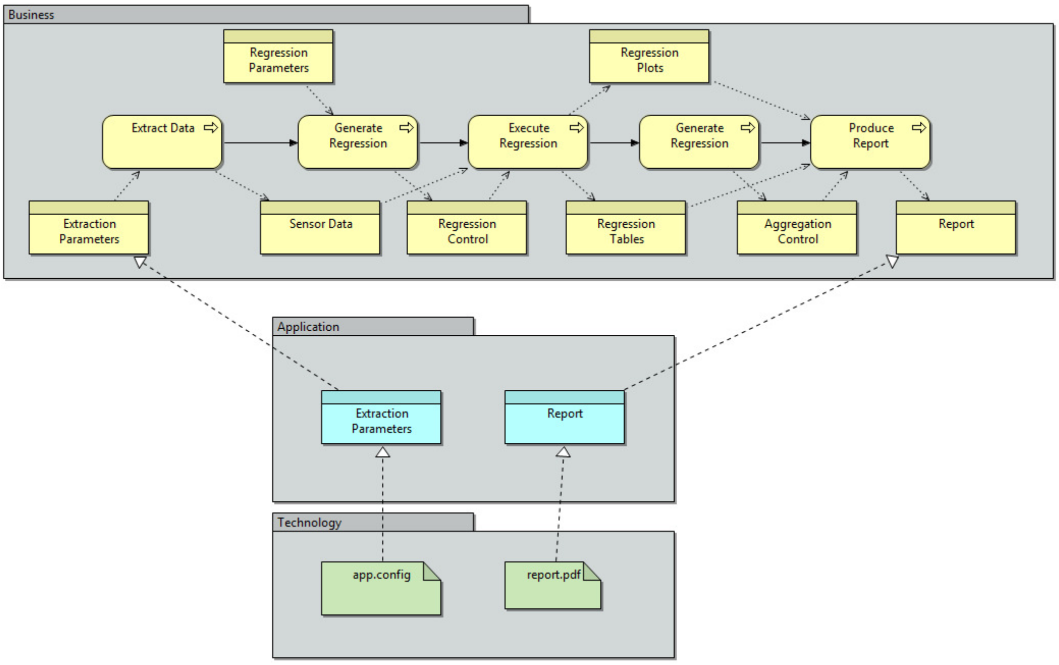
Metric 1 – Identical PDFs

Description: Any PDF report generated (using the same extraction and regression parameters) in the redeployment environment must be identical according to the following aspects: number of pages, number of sections, number of figures, number of tables and words.

Target Operator: Equal

Target Value: NA

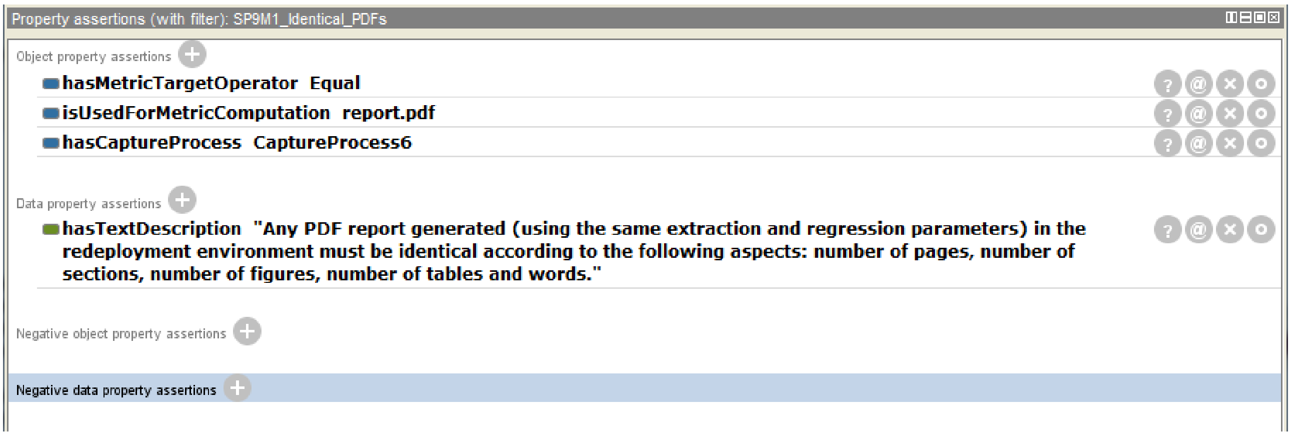
Capture Process:



Description: Execute LNEC2 Process and get the PDF report as output.

The measurement points were defined directly in the VPlan by linking elements of the capture processes with corresponding elements of the preserved process. We have also specified which items will be used for metrics computation.

Figure 25 presents the metric SP9M1 (Metric 1 of the Significant Property SP9) used here as an example. As illustrated, the produced report (report.pdf) will be stored and used for metric computation. There is no target value defined, hence the original response and the one intercepted in the redeployment environment has to be compared. They are supposed to be equal in order to be verified positively. Furthermore, in the Figure one can see the description of the metric and also that the metric has one instance of a capture process. The name (CaptureProcess6) implies that it is an instance of a capture process 6.



Property assertions (with filter): SP9M1_Identical_PDFs

Object property assertions

hasMetricTargetOperator Equal

isUsedForMetricComputation report.pdf

hasCaptureProcess CaptureProcess6

Data property assertions

hasTextDescription "Any PDF report generated (using the same extraction and regression parameters) in the redeployment environment must be identical according to the following aspects: number of pages, number of sections, number of figures, number of tables and words."

Negative object property assertions

Negative data property assertions

Figure 25: Example of a metric modelled in VPlan and accessed using Protégé.

3.7.2.4 VFramework step 4 – Capture verification data

Considering the defined metrics to assess the significant properties in the MLR process the capture of verification data comprised of the execution of the full process and storage of the data files necessary for metric computation. The exceptions were:

- Metrics from the SP1 entitled “Generate Data” that required a manual count of data files and number of lines in data files; and
- Metric SP9-M1 entitled “Identical PDFs” were we used a self-developed tool to extract necessary properties (e.g. number of pages, number of words, number of sections, etc) from a PDF file (report.pdf)

We have used the VHelper (see Section 3.5) to generate the folder structure to which we have copied the collected data. Through the VHelper the data was automatically added to the ontology. VHelper has also validated the data.

In the last step we have executed a set of SPARQL queries (see Section 3.6 and Annex A) to validate if the VPlan has all necessary information and if the required concepts are defined. This was the last step of the VFramework in the original environment. Thus we have created evidence allowing verifying the redeployed process.

3.7.2.5 VFramework step 5 – Prepare system for redeployment

Since the MLR process depends on Microsoft .NET Framework 4.0 which runs Windows on the original environment we opted to test the redeployment scenario in a machine running Ubuntu Linux[12] 12.10. Ubuntu Linux is an open source, which is based on the GNU Linux kernel.

In order to install the application components (TIMBUSclient) that depend on .NET Framework 4.0 we ported the client application into a Wine[13] environment. Wine allows running Windows application on

[12] www.ubuntu.om

[13] www.winehq.org

different operating systems and was able to run the TIMBUS client component. For the execute regression activity it was required to install the R[14] Project for statistical computing package and a Latex compiler (in this example we used TexLive[15]).

3.7.2.6 VFramework step 6 – Capture redeployment performance data

The redeployment performance data must be captured for all metrics. According to the design verification settings defined in section 3.7.2.3, the capture process was executed for all metrics. This included restoring the configuration data that was present in the original environment. Similarly to step 4 entitled "capture verification data" (section 3.7.2.4), the target value of the capture process in the redeployment environment can be computed by automatic or manual means, i.e.:

- Metrics from the SP1 entitled "Generate Data" required a manual count of data files and number of lines in data files;
- Metric SP9-M1 entitled "Identical PDFs" used a self-developed tool to extract necessary properties (e.g. number of pages, number of words, number of sections, etc) from a PDF file (report.pdf); and
- The remaining metrics were manually computed.

3.7.2.7 VFramework step 7 – Compare and Assess

Depending on the type of metric and target operator it might be needed to compare the results of capture process in the redeployment environment with capture verification data collected in step 4 (section 3.7.2.4). Also, comparison can be done manual or with the help of automatic tools. Table 3 describes, for each metric, the type of comparison that was performed and which tools (if used) were used for the comparison.

Table 3: Metric Assessment in the MLR process.

Metric	Comparison with original data files?	Tool Support
SP1-M1	Yes	Comparison was manual.
SP1-M2	Yes	Comparison was manual.
SP2-M1	No	Comparison was manual.
SP2-M2	No	Comparison was manual.

[14] www.r-project.org

[15] www.tug-org/texlive/

SP2-M3	No	Comparison was manual.
SP3-M1	No	Supported by the Latex and PNG comparator (see Annex C for more details).
SP4-M1	No	Supported by the Latex comparator (see Annex C for more details).
SP5-M1	No	Supported by the Latex comparator (see Annex C for more details).
SP6-M1	No	Supported by the Latex and PNG comparator (see Annex C for more details).
SP7-M1	No	Supported by the Latex comparator (see Annex C for more details).
SP8-M1	No	Supported by the PNG comparator (see Annex C for more details).
SP9-M1	Yes	Supported mainly with the PDF comparator. PNG comparator could also be used for images embedded into the PDF (see Annex C for more details).

Having performed the comparison process we conclude that the redeployment was successful, i.e. the redeployed process instance performs correctly in the new environment.

3.8 Use case: application on Open Source Workflow

This section presents application of the developed concepts on a use case. We have selected a Taverna[16] workflow from one of the open source workflows introduced by the use cases in WP7 (TIMBUS Consortium, 2013c). It has been selected, because it allows us to illustrate how verification information for a complex process can be collected. The process uses external third-party services, depends on various libraries inside and outside of the workflow engine and therefore makes the case interesting. In the consecutive subsections we will present the use case and describe how the proposed verification solution is applied.

For this use case, the description will focus only on the phase conducted in the original environment and hence only four steps of the VFramework are described. This phase can be considered to be more important during the VFramework application, because only if this phase was correctly executed, the later verification in the redeployment environment is possible. Furthermore, the first phase is more complex and requires more consideration when collecting the evidence. Several influencers and dependencies have to be discovered and data collection processes have to be defined. The second phase of the VFramework reuses already available information and is guided to a great extent by the information collected in the original environment. Therefore we decided to describe the first phase on a different use case to demonstrate once more how the VFramework should be applied.

[16] Taverna: http://www.taverna.org.uk/

3.8.1 Use case description

The description provided in this section is based on information from the deliverable D7.7 Preservation of Open Source Worklfows (TIMBUS Consortium, 2013c). For more information please refer to this deliverable.

The selected workflow, named “Validate Wav File Format using JHOVE2 SCAPE Web Service Workflow”, is available at myExperiment at http://www.myexperiment.org/workflows/2637.html, under the Creative Commons Attribution Share Alike 3.0 License.

“The workflow illustrates file format identification using the JHOVE[17] tool. JHOVE provides methods to perform format-specific identification, validation, and characterization of digital objects. File characterization is an important step when performing digital preservation, to get an overview of the formats that need to be preserved and to set appropriate preservation actions.

The Taverna[18] model of the workflow is depicted in Figure 26. The workflow has one input, the location of the file to be analysed. Although the workflow’s name suggests that it is only suitable for WAVE files, the characterisation tool utilised and thus also the workflow implemented, can characterize any kind of file format known to JHOVE2. The process has several different outputs. Most interesting is the format name identified; the remaining outputs are meta-data on the characterisation process, such as the time needed to run, whether it was successful, or a descriptive message.

The main step of the workflow is the said call to the characterisation service; the other steps are mostly concerned with preparing services invocation and parsing the output into separate pieces. The workflow characterisation step is provided by a call to an externally available web service, which provides the JHOVE file characterisation functionality” (TIMBUS Consortium, 2013c).

[17] JHOVE: http://www.dcc.ac.uk/resources/external/jstorharvard-object-validation-environment-jhove

[18] Taverna: http://www.taverna.org.uk/

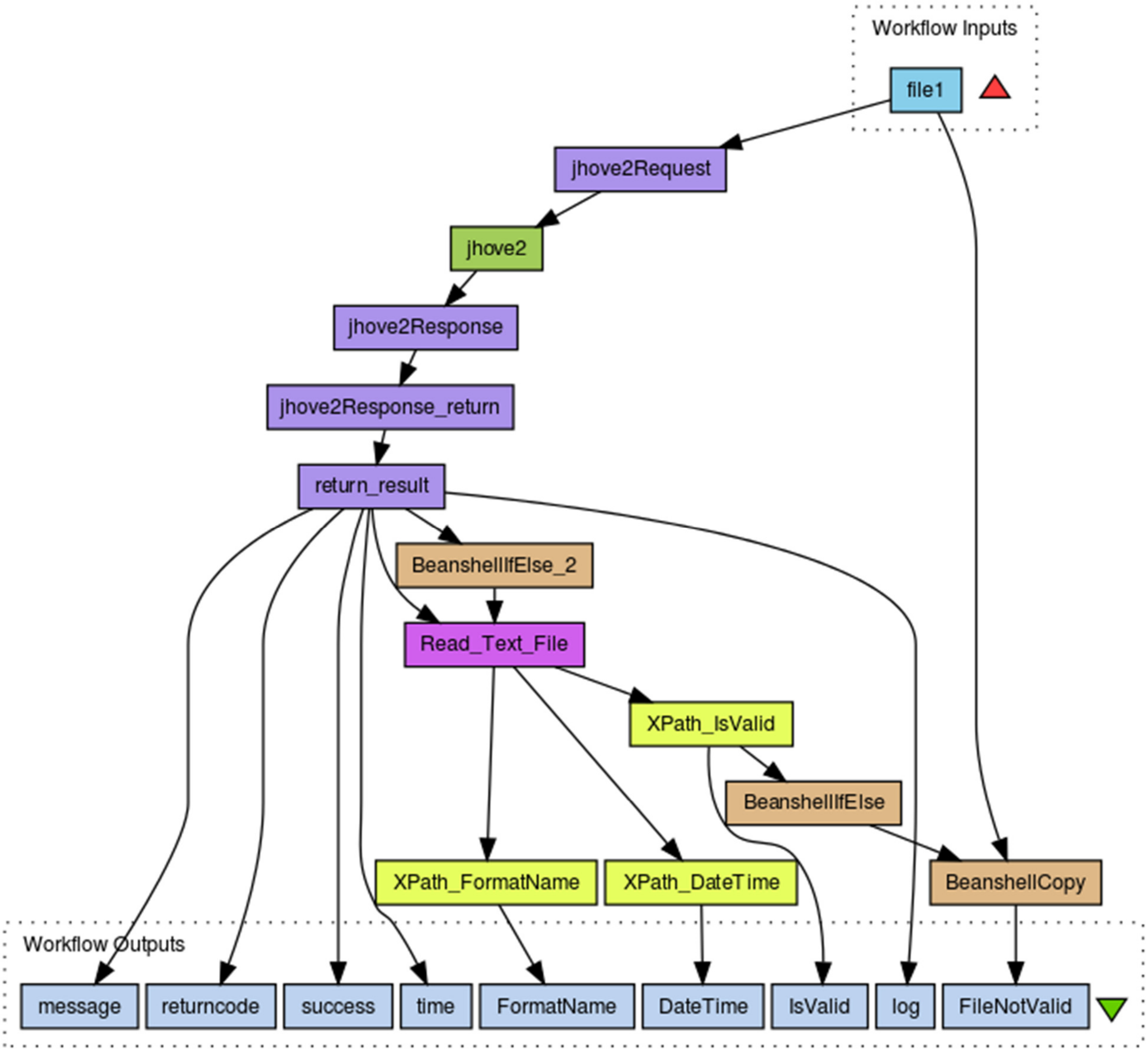


Figure 26: Taverna workflow for file characterisation.

3.8.2 VFramework application

We will explain the verification process performed in the original environment by showing how each of the VFramework steps was executed.

3.8.2.1 VFramework step 1 – Describe the original environment

According to the Section 3.4.3 we have initialized a clean ontology file in the Protégé ontology editor and provided basic information about the authors. Then for the description of the process we have used the context model of the process modelled in Archimate. In the future, when TIMBUS extractors and converters will be fully implemented, the context model will be automatically created basing on the workflow specification. The Figure 27 depicts the context model of the workflow. The context model has been converted to OWL format and imported into the VPlan.

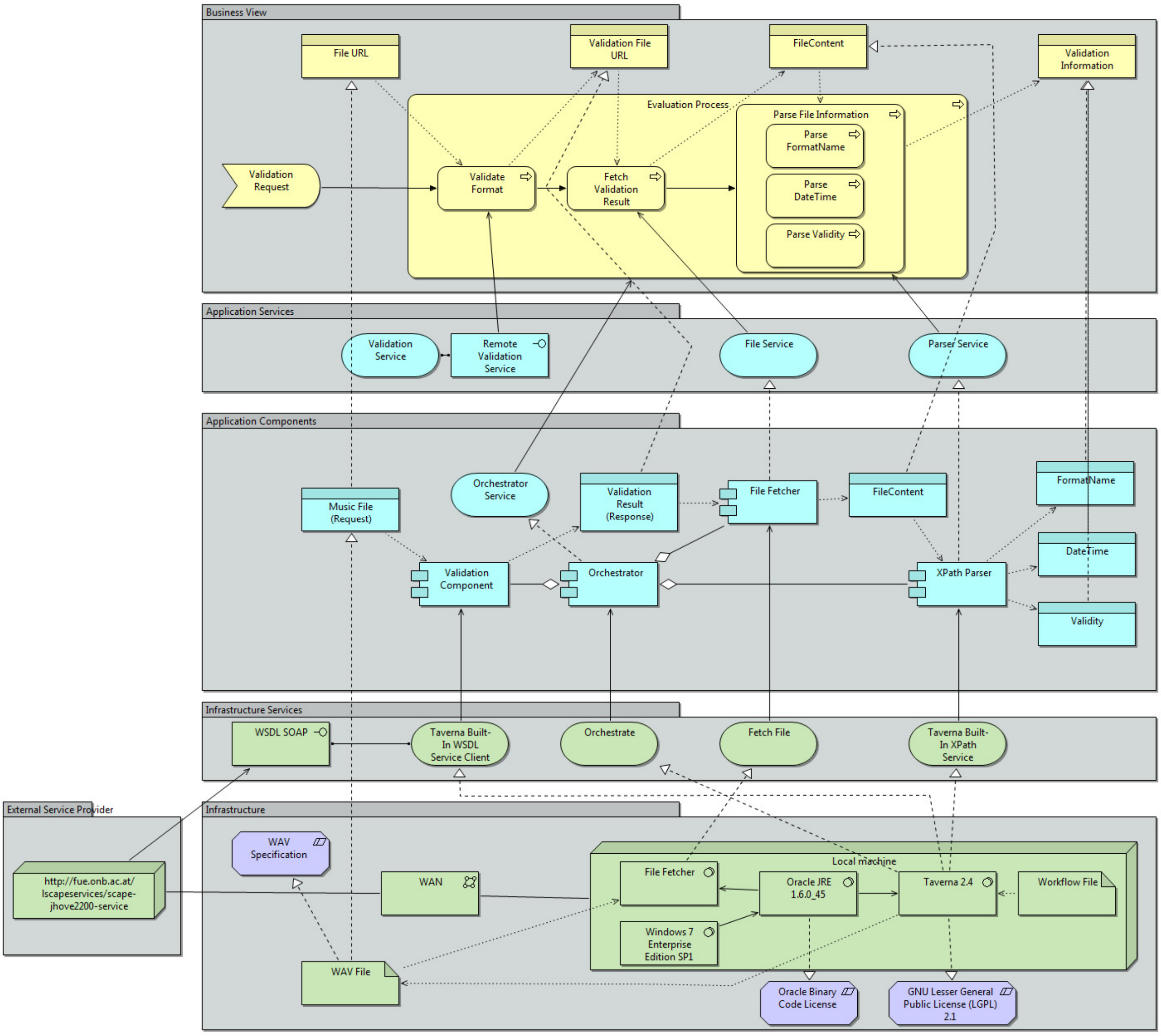


Figure 27: ArchiMate model of the workflow.

Then we have defined one redeployment scenario for this workflow, which assumes that the process will be rerun in the future including all its steps. We assume that the scenario will be redeployed in order to perform characterization of files in the same way as it used to do in the original environment. The scenario allows substituting steps of the process if needed.

We have selected one instance used for verification. The instance consists only of input file provided to the workflow. This is the file which is provided by default by the author of the workflow.

The last action was specification of significant properties which are presented in Table 4. Each of them has its id, name and description.

Table 4: Significant properties defined for the workflow case.

ID: SP1	**Name:** URL referenced WAVE files
Short description: The workflow processes correctly the WAVE files which location is specified as URL.	
Additional information: none	
ID: SP2	**Name:** Execution sequence
Short description: The workflow steps are executed in the order conforming to the workflow specification.	
Additional information: Steps of the workflow cannot be executed in arbitrary order. They have to conform to the workflow specification.	
ID: SP3	**Name:** Correct outputs
Short description: The workflow delivers results appropriately to the workflow specification.	
Additional information: none	
ID: SP4	**Name:** Web Service Requests similarity
Short description: The request sent to the web service conforms to the WSDL file of the service.	
Additional information: The WSDL file is in the same version as during the capture. The WS used during the redeployment also conforms to this WSDL file version.	
ID: SP5	**Name:** Web Service Responses similarity
Short description: The result file obtained from the WS conforms to the WSDL specification.	
Additional information: none	
ID: SP6	**Name:** Result file correctness
Short description: The result file obtained from the WS has similar information.	
Additional information: none	
ID: SP7	**Name:** Result file validity
Short description: The JHOVE result file is a valid XML document	
Additional information: none	
ID: SP8	**Name:** Wave features equal
Short description: The features of the JHOVE result file describe the WAVE format	

Additional information: The JHOVE result file contains many modules providing additional information not related to the characterisation of the file, e.g. the configuration of the system running the WS, paths to temporary file locations, etc. In the considered scenario, the system configuration of the WS does not have to match the original one. Therefore, only the metrics related to WAVE format detection are in focus.	
ID: SP9	**Name:** Execution time similarity
Short description: The execution time does not exceed by an order of magnitude the original execution time.	
Additional information: none	

All of the information collected at this step has been added to the VPlan. The state of the VPlan after execution of the first step is depicted in Figure 28. In order to enhance readability the elements of the preserved process which was imported to the VPlan are not depicted.

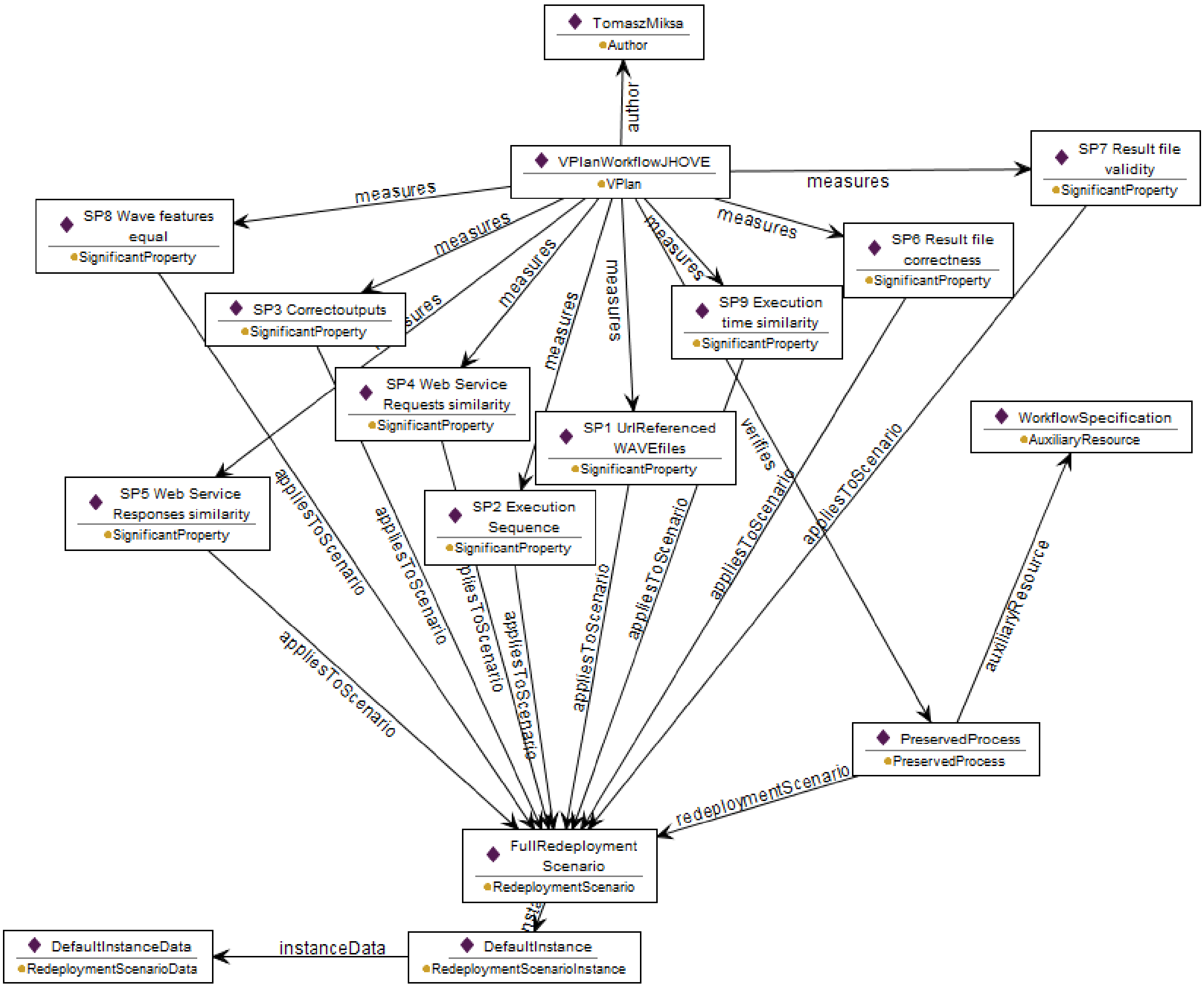


Figure 28: Simplified visualisation of the VPlan after the first step of the VFramework.

3.8.2.2 VFramework step 2 – Prepare system for preservation

We have analysed the workflow and its environment and checked if the context model created in the first step has any specific dependencies or deterministic issues. The analysis has revealed that the workflow is deterministic and has one external dependency which is a call to the external web service. This web service must be in place in order to re run the workflow. It is beyond the scope of the verification task to decide whether the web service will be available during the redeployment or if it also has to be preserved. The VFramework states that this web service is needed and therefore is present in the context model. No changes to the VPlan were necessary at this step.

3.8.2.3 VFramework step 3 – Design verification setting

In the third step of the VFramework we have assigned metrics to the significant properties. In some cases more than one metric is used to measure the significant property, while in other cases the same metric can be used to measure two or more significant properties. Furthermore, for each of the metrics we have assigned their target operators and target values which will be used for the assessment during the redeployment. For each of the metrics a capture process was assigned. In some cases the same capture process is used for different metrics. Table 5 provides overview of information which was gathered at this step. Figure 29, Figure 30, Figure 31, and Figure 32 depict the capture processes. All this information was stored in the VPlan.

Table 5: Significant properties, metrics and capture processes of the workflow use case.

ID: SP1	**Name:** URL referenced WAVE files
Short description: The workflow processes correctly the WAVE files which location is specified as URL.	
Additional information: none	
Metric SP1M1 – URL referenced WAVE file is a valid input	
Description: The workflow accepts as a valid input the WAVE files which location is specified using URLs.	
Target Operator: Equal	
Target Value: The output 'FormatName' equals WAVE and the 'returncode' success output equals true.	
Capture Process: [CP2] Run the workflow and save the provenance data. The provenance data is a zip file which has an ontology file with details of the execution and also folders in which the input and the output data of the steps are stored. For the verification process, one needs to open this ontology file in Protégé and find the individuals of ProcessRun. Also the folders containing outputs may need to be inspected.	
ID: SP2	**Name:** Execution sequence
Short description: The workflow steps are executed in the order conforming to the workflow specification.	

<table>
<tr><td colspan="2"><b>Additional information</b>: Steps of the workflow cannot be executed in arbitrary order. They have to conform to the workflow specification.</td></tr>
<tr><td colspan="2"><b>Metric SP2M1–</b> the execution sequence is correct</td></tr>
<tr><td colspan="2"><b>Description:</b> The original and the redeployed workflow execute workflow steps in the same order.</td></tr>
<tr><td colspan="2"><b>Target Operator:</b> Equals</td></tr>
<tr><td colspan="2"><b>Target Value:</b> steps sequence file</td></tr>
<tr><td colspan="2"><b>Capture Process:</b>
[CP1] Provenance trace has data property ‘startedAtTime’ in ‘Process Run’ individuals. By parsing the trace a sequence of steps can be obtained and compared to the original trace. Only the sequence is important, not the actual time.</td></tr>
<tr><td><b>ID</b>: SP3</td><td><b>Name:</b> Correct outputs</td></tr>
<tr><td colspan="2"><b>Short description</b>: The workflow delivers results appropriately to the workflow specification.</td></tr>
<tr><td colspan="2"><b>Additional information</b>: none</td></tr>
<tr><td colspan="2"><b>Metric SP3M1 –</b> time output provides numerical value expressing WS execution time</td></tr>
<tr><td colspan="2"><b>Description:</b> Time output provides numerical value expressing WS execution time. It is not the workflow execution time. The WS execution time should be lower or equal than the workflow execution time. The output contains only a single number consisting of digits [0-9]. The value is an integer expressed in milliseconds.</td></tr>
<tr><td colspan="2"><b>Target Operator:</b> Lower or equal</td></tr>
<tr><td colspan="2"><b>Target Value:</b> Workflow execution duration (also obtained from the provenance trace)</td></tr>
<tr><td colspan="2"><b>Capture Process:</b>
[CP2] Extract outputs of the provenance trace of the workflow.</td></tr>
<tr><td colspan="2"><b>Metric SP3M2 –</b> success output provides Boolean values corresponding to the correctness of WS execution</td></tr>
<tr><td colspan="2"><b>Description:</b> Verification of the success output port of the workflow.</td></tr>
<tr><td colspan="2"><b>Target Operator:</b> Equals</td></tr>
<tr><td colspan="2"><b>Target Value:</b> The value is tested for being either true or false. Other outputs don’t contain errors (provenance trace files stored in the output folder don’t have .err extension)</td></tr>
<tr><td colspan="2"><b>Capture Process:</b>
[CP2] Extract outputs of the provenance trace of the workflow.</td></tr>
</table>

Metric SP3M3 – return code output provides valid return code from the WS
Description: Verification of the return code output port of the workflow.
Target Operator: Equals
Target Value: The output contains an integer value which is either 0 or 1. It is 0 when the output success is true and other outputs don't contain errors. It is 1 when output success is false and other outputs don't contain errors.
Capture Process: [CP2] Extract outputs of the provenance trace of the workflow.
Metric SP3M4 – message provides a valid string returned from the WS
Description: Verification of the message output port of the workflow.
Target Operator: Equals
Target Value: The value must be string containing: "There should stand something interesting."
Capture Process: [CP2] Extract outputs of the provenance trace of the workflow.
Metric SP3M5 – isValid output provides information on validity of the file according to JHOVE
Description: Verification of the isValid output port of the workflow.
Target Operator: Equals
Target Value: The value is tested for being either true or false. If the input WAVE file is a valid WAVE file according to its specification, then the value is true, otherwise false.
Capture Process: [CP2] Extract outputs of the provenance trace of the workflow.
Metric SP3M6 – DateTime output provides information on a date and time when the JHOVE WS started processing the file
Description: Verification of the DateTime output port of the workflow.
Target Operator: Higher or equal
Target Value: The value is in format yyyy-MM-dd'T'HH:mm:ssZ. The value is later or equal than the date and time of process execution start.
Capture Process: [CP2] Extract outputs of the provenance trace of the workflow.

Metric SP3M7 – FileNotValid output provides filename of the input file if the file format is not recognized by the WS	
Description: Verification of the FileNotValid output port of the workflow.	
Target Operator: Equals	
Target Value: If the output isValid equals true then the FileNotValid is empty. In other case, the output contains the filename of the input file.	
Capture Process: [CP2] Extract outputs of the provenance trace of the workflow.	
ID: SP4	**Name:** Web Service Requests similarity
Short description: The request sent to the web service conforms to the WSDL file of the service.	
Additional information: The WSDL file is in the same version as during the capture. The WS used during the redeployment also conforms to this WSDL file version.	
Metric SP4M1 – requests are similar	
Description: The requests in both environments are the same. (Header information does not need to be exact)	
Target Operator: Equals	
Target Value:	
Capture Process: [CP4] Intercept the SOAP query of the verified process.	
ID: SP5	**Name:** Web Service Responses similarity
Short description: The result file obtained from the WS conforms to the WSDL specification.	
Additional information: none	
Metric SP5M1 – responses are similar	
Description: The responses in both environments are the same.	
Target Operator: Equals	
Target Value:	
Capture Process: [CP3] Intercept the SOAP response of the verified process.	
ID: SP6	**Name:** Result file correctness

Short description: The result file obtained from the WS has similar information.	
Additional information: none	
Metric SP3M1 – time provides numerical value expressing WS execution time	
Metric SP3M2 – success provides Boolean values corresponding the correctness of WS execution	
Metric SP3M3 – returncode output provides valid return code from the WS	
Metric SP3M4 – message provides a valid string returned from the WS	
ID: SP7	**Name:** Result file validity
Short description: The JHOVE result file is a valid XML document	
Additional information: none	
Metric SP7M1 – JHOVE XML is valid	
Description: The XML file returned from the web service is validated.	
Target Operator: Equals	
Target Value: The XML file returned from the web service is validated with an XML validation tools against the JHOVE XSD schema. If there are no errors, the verification is positive.	
Capture Process: [CP2] Extract outputs of the provenance trace of the workflow.	
ID: SP8	**Name:** Wave features equal
Short description: The features of the JHOVE result file describe the WAVE format	
Additional information: The JHOVE result file contains many modules providing additional information not related to the characterisation of the file, e.g. the configuration of the system running the WS, paths to temporary file locations, etc. In the considered scenario, the system configuration of the WS does not have to match the original one. Therefore, only the metrics related to WAVE format detection are in focus.	
Metric SP8M1 – WAVEModule features are the same	
Description: The original and the new JHOVE results file are compared. Both files must contain identical set of features for http://jhove2.org/terms/reportable/org/jhove2/module/format/wave/WAVEModule	
Target Operator: Equals	
Target Value: NA	
Capture Process: [CP2] Extract outputs of the provenance trace of the workflow	

ID: SP9 **Name:** Execution time similarity

Short description: The execution time does not exceed by an order of magnitude the original execution time.

Additional information: none

Metric SP9M1– execution duration

Description: The execution duration of the redeployed workflow is not higher than 10 times duration of the original workflow.

Target Operator: Lower or equal

Target Value: 10 times duration of the original workflow

Capture Process:

[CP2] Extract outputs of the provenance trace of the workflow.

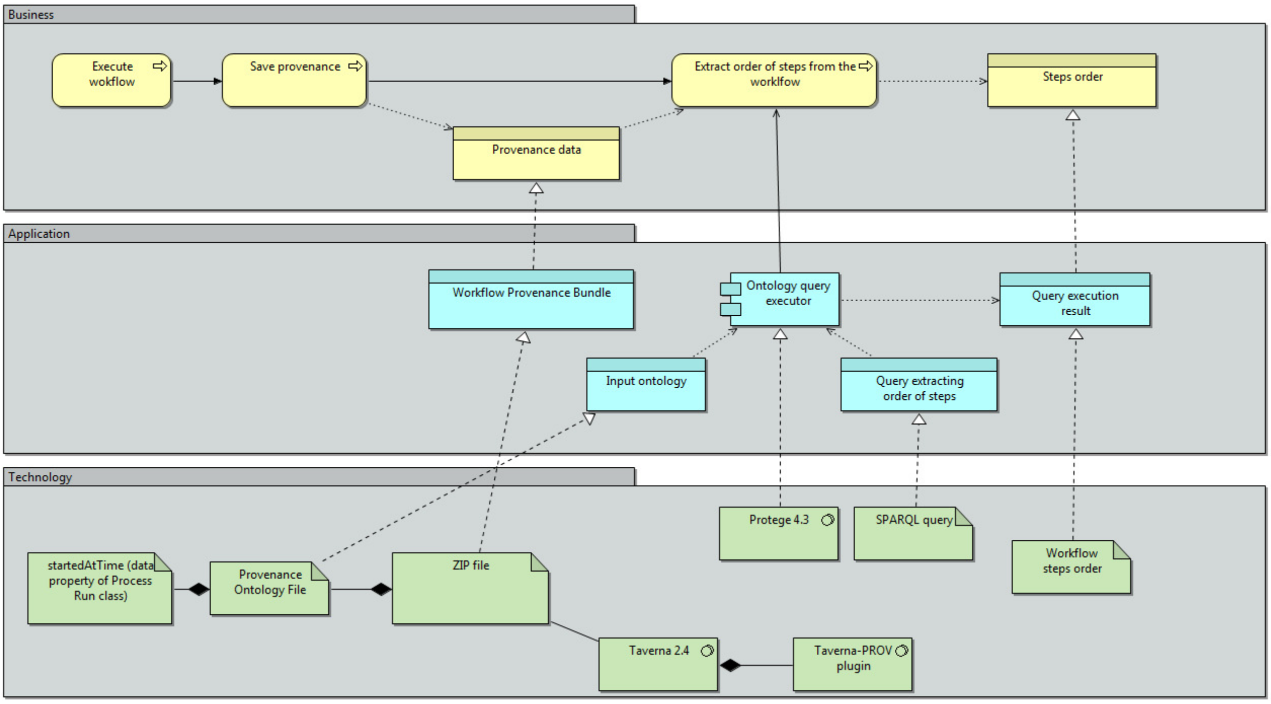


Figure 29: Capture Process CP1.

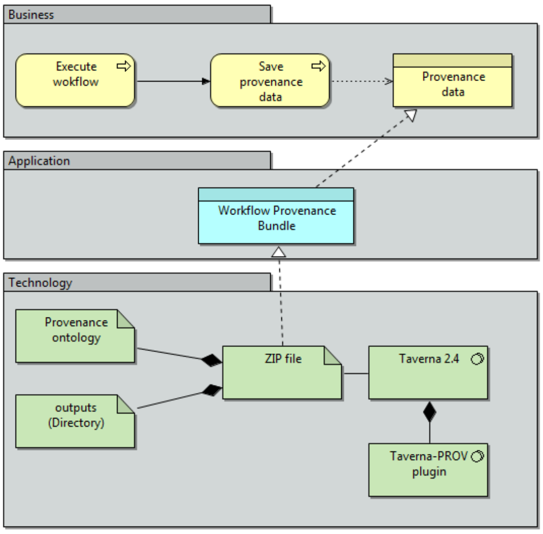


Figure 30: Capture Process CP2.

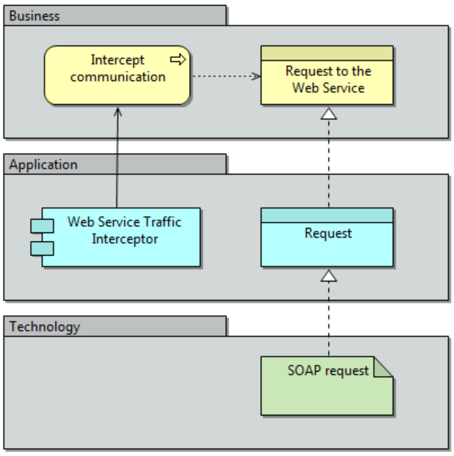


Figure 31: Capture Process CP3.

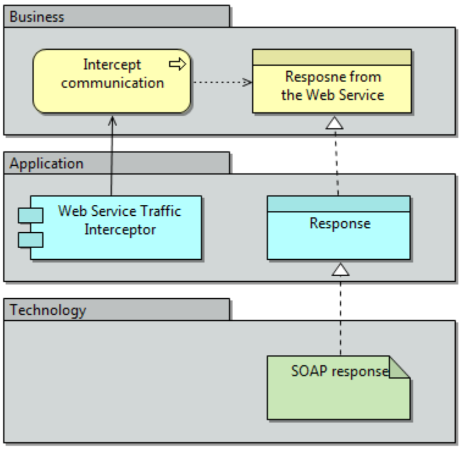
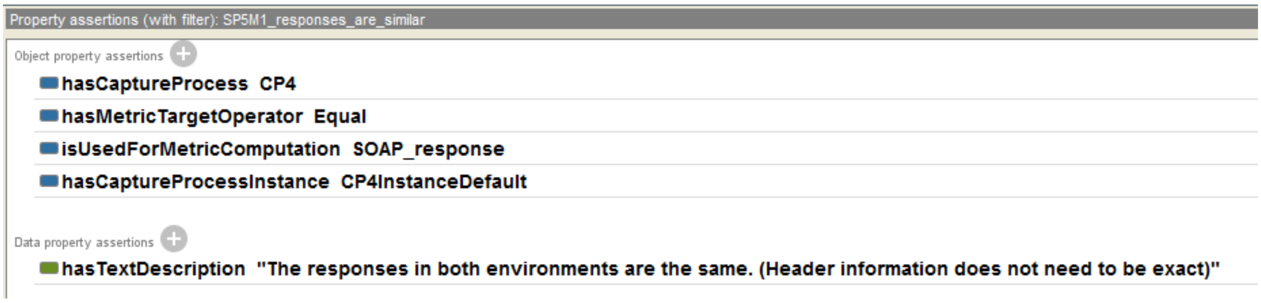


Figure 32: Capture Process CP4.

The measurement points were defined directly in the VPlan by linking elements of the capture processes with corresponding elements of the preserved process. We have also specified which items will be used for metrics computation.

Figure 33 presents the metric SP5M1 used here as an example. According to it, the metric has capture process CP4. The *SOAP_response* (part of CP4) will be stored and used for metric computation. There is no target value defined, hence the original response and the one intercepted in the redeployment environment has to be compared. They are supposed to be equal in order to be verified positively. Furthermore, in the Figure one can see the description of the metric and also that the metric has one instance of a capture process. The name implies that it is an instance of a capture process CP4 for a default instance of the redeployment scenario.


Property assertions (with filter): SP5M1_responses_are_similar

Object property assertions

- hasCaptureProcess CP4
- hasMetricTargetOperator Equal
- isUsedForMetricComputation SOAP_response
- hasCaptureProcessInstance CP4InstanceDefault

Data property assertions

- hasTextDescription "The responses in both environments are the same. (Header information does not need to be exact)"

Figure 33: Example of a metric modelled in VPlan and accessed using Protégé.

3.8.2.4 VFramework step 4 – Capture verification data

According to the requirements of the capture processes we have installed a provenance plugin for Taverna which allows exporting provenance traces consisting of ontology describing workflow execution and data exchanged between workflow steps. Furthermore, we have used WireShark[19] to intercept the communication to and from the web service. Having prepared the tools for capturing we have executed the workflow providing the input data according to the definition of the redeployment scenario instance. The data has been successfully captured. The next step was its addition to the VPlan.

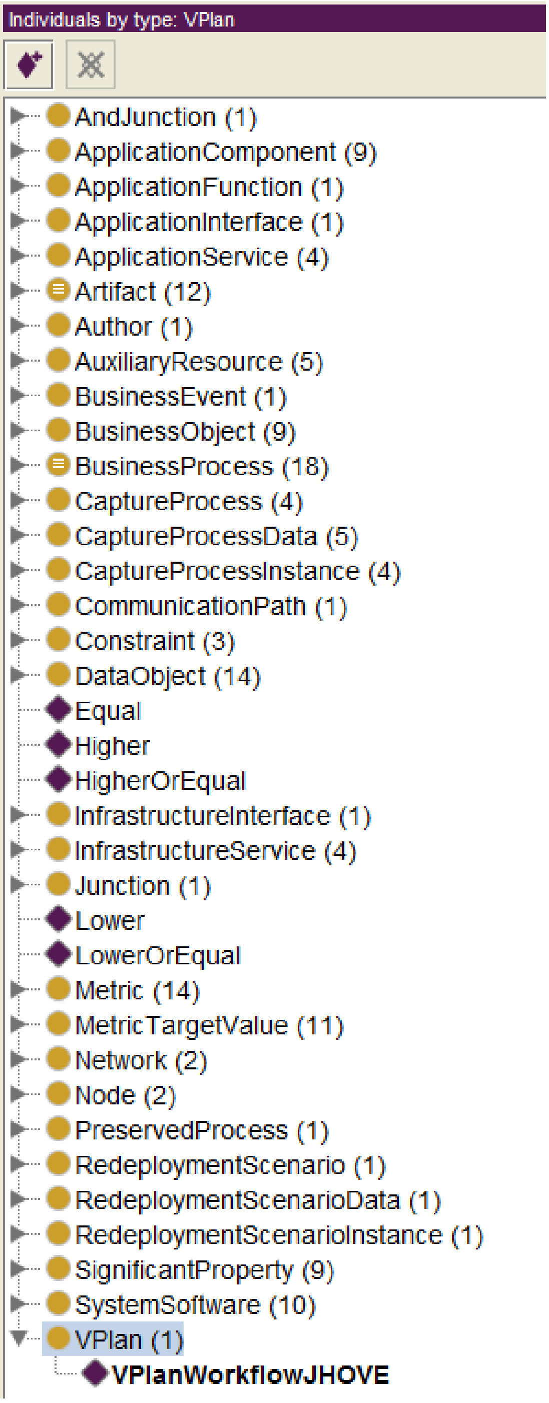
We have used the VHelper (see Section 3.5) to generate the folder structure to which we have copied the collected data. The folder structure is depicted in Figure 16. Then we copied manually the provenance trace to the appropriate folder. The intercepted request and response were exported from the WireShark into a text file and also copied to a corresponding location and automatically added to the ontology and validated the data using the VHelper tool. Finally, we have inspected the VPlan using Protégé and renamed some of the automatically generated labels to more human readable form.

In the last step we have executed a set of SPARQL queries (see Section 3.6 and Annex A) to validate if the VPlan has all necessary information and if the required concepts are defined. Some missing object properties were fixed and the VPlan was positively validated. Figure 34 presents a summary of classes and

[19] WireShark: http://www.wireshark.org/

individuals created during the first phase of the VFramework application. Figure 35 depicts a directory listing of the data folder in which the verification data was stored.

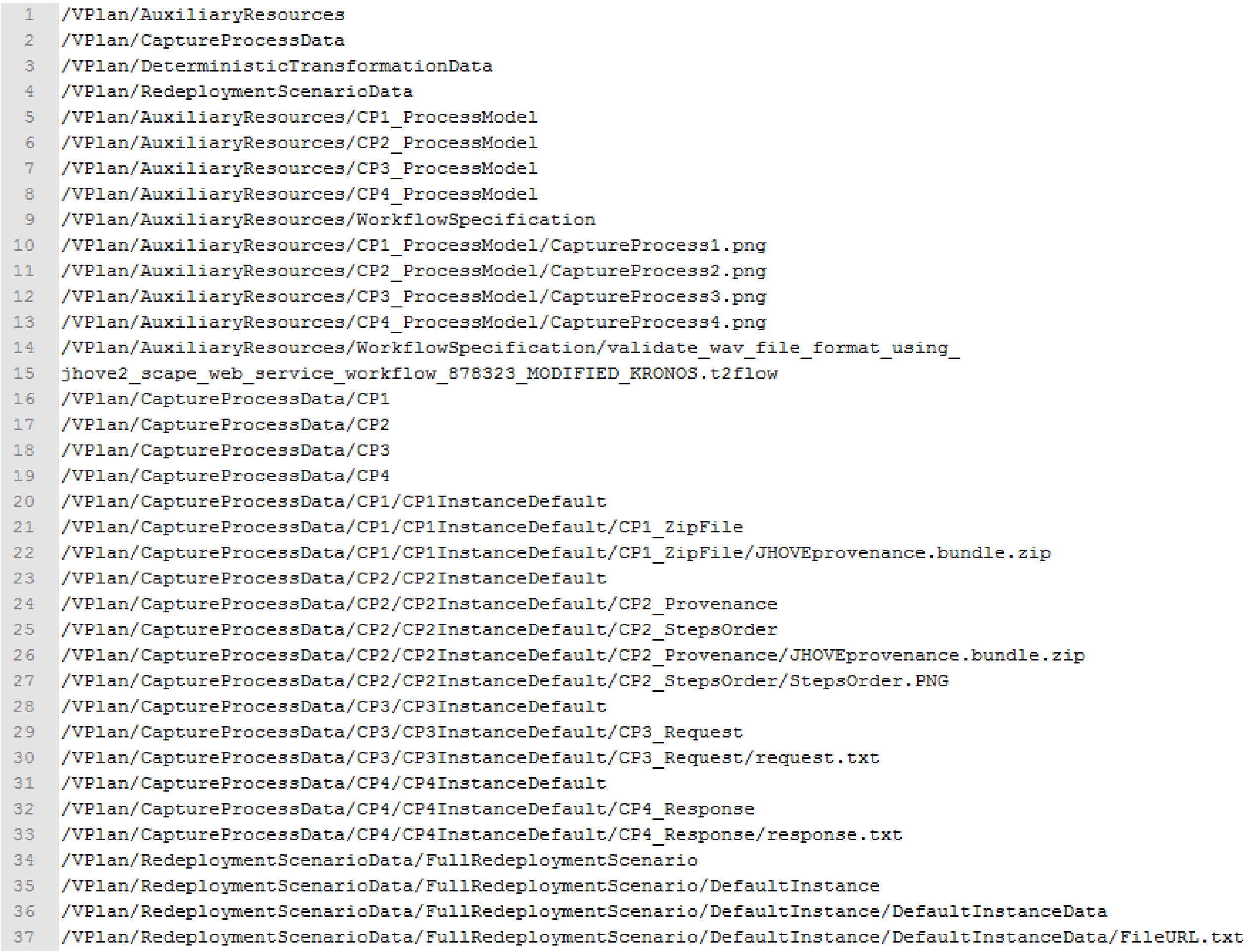
This was the last step of the VFramework in the original environment. Thus we have created evidence allowing verifying the redeployed process.



Individuals by type: VPlan

- AndJunction (1)
- ApplicationComponent (9)
- ApplicationFunction (1)
- ApplicationInterface (1)
- ApplicationService (4)
- Artifact (12)
- Author (1)
- AuxiliaryResource (5)
- BusinessEvent (1)
- BusinessObject (9)
- BusinessProcess (18)
- CaptureProcess (4)
- CaptureProcessData (5)
- CaptureProcessInstance (4)
- CommunicationPath (1)
- Constraint (3)
- DataObject (14)
- Equal
- Higher
- HigherOrEqual
- InfrastructureInterface (1)
- InfrastructureService (4)
- Junction (1)
- Lower
- LowerOrEqual
- Metric (14)
- MetricTargetValue (11)
- Network (2)
- Node (2)
- PreservedProcess (1)
- RedeploymentScenario (1)
- RedeploymentScenarioData (1)
- RedeploymentScenarioInstance (1)
- SignificantProperty (9)
- SystemSoftware (10)
- VPlan (1)
  - **VPlanWorkflowJHOVE**

Figure 34: Summary of the VPlan created for the workflow use case. The list presents classes and the value in brackets denotes a number of individuals of each class.



```
/VPlan/AuxiliaryResources
/VPlan/CaptureProcessData
/VPlan/DeterministicTransformationData
/VPlan/RedeploymentScenarioData
/VPlan/AuxiliaryResources/CP1_ProcessModel
/VPlan/AuxiliaryResources/CP2_ProcessModel
/VPlan/AuxiliaryResources/CP3_ProcessModel
/VPlan/AuxiliaryResources/CP4_ProcessModel
/VPlan/AuxiliaryResources/WorkflowSpecification
/VPlan/AuxiliaryResources/CP1_ProcessModel/CaptureProcess1.png
/VPlan/AuxiliaryResources/CP2_ProcessModel/CaptureProcess2.png
/VPlan/AuxiliaryResources/CP3_ProcessModel/CaptureProcess3.png
/VPlan/AuxiliaryResources/CP4_ProcessModel/CaptureProcess4.png
/VPlan/AuxiliaryResources/WorkflowSpecification/validate_wav_file_format_using_
jhove2_scape_web_service_workflow_878323_MODIFIED_KRONOS.t2flow
/VPlan/CaptureProcessData/CP1
/VPlan/CaptureProcessData/CP2
/VPlan/CaptureProcessData/CP3
/VPlan/CaptureProcessData/CP4
/VPlan/CaptureProcessData/CP1/CP1InstanceDefault
/VPlan/CaptureProcessData/CP1/CP1InstanceDefault/CP1_ZipFile
/VPlan/CaptureProcessData/CP1/CP1InstanceDefault/CP1_ZipFile/JHOVEprovenance.bundle.zip
/VPlan/CaptureProcessData/CP2/CP2InstanceDefault
/VPlan/CaptureProcessData/CP2/CP2InstanceDefault/CP2_Provenance
/VPlan/CaptureProcessData/CP2/CP2InstanceDefault/CP2_StepsOrder
/VPlan/CaptureProcessData/CP2/CP2InstanceDefault/CP2_Provenance/JHOVEprovenance.bundle.zip
/VPlan/CaptureProcessData/CP2/CP2InstanceDefault/CP2_StepsOrder/StepsOrder.PNG
/VPlan/CaptureProcessData/CP3/CP3InstanceDefault
/VPlan/CaptureProcessData/CP3/CP3InstanceDefault/CP3_Request
/VPlan/CaptureProcessData/CP3/CP3InstanceDefault/CP3_Request/request.txt
/VPlan/CaptureProcessData/CP4/CP4InstanceDefault
/VPlan/CaptureProcessData/CP4/CP4InstanceDefault/CP4_Response
/VPlan/CaptureProcessData/CP4/CP4InstanceDefault/CP4_Response/response.txt
/VPlan/RedeploymentScenarioData/FullRedeploymentScenario
/VPlan/RedeploymentScenarioData/FullRedeploymentScenario/DefaultInstance
/VPlan/RedeploymentScenarioData/FullRedeploymentScenario/DefaultInstance/DefaultInstanceData
/VPlan/RedeploymentScenarioData/FullRedeploymentScenario/DefaultInstance/DefaultInstanceData/FileURL.txt
```

Figure 35: Directory listing of a VPlan data folder in which verification data was stored.

4 Security

In order to plan, execute and assess preservation activities relating to the security of the business processes an operationalization of security has to be conducted as to derive actionable tasks for the phases and activities of the TIMBUS preservation workflow (TIMBUS Consortium, 2013b).

The following sections aim to establish such an operationalization by providing a domain-specific ontology (DSO) (see (TIMBUS Consortium, 2013a)) for security, i.e. a security ontology. Before such an ontology can be developed, first the security principles and features that impact preservation are discussed. Secondly, these impacts are reviewed in more detail to understand the requirements for a security ontology in the context of TIMBUS and related work's fitness for this purpose is assessed. Then we present the concepts that we included into the ontology and describe their properties and relationships in detail in Section 4.1 and Section 4.2.

4.1 Security Principles

Information has become the key asset for modern enterprises as it contributes a significant value to our society, thus protection from unauthorized access is required. The following sections describe the fundamental concepts that are relevant for the security ontology and provide an overview of the implications of preserving security features of business processes for the TIMBUS framework (TIMBUS Consortium, 2013b).

4.1.1 Quality Models and Security

In information technology, security is considered as part of the overall notion of product quality. Here, product quality denotes the "[…] degree to which the system satisfies the stated and implied needs of its various stakeholders, and thus provides value" (ISO/IEC, 2010, S. 10). In order to render this notion of quality more tangible, quality models subdivide this notion of quality into more granular quality characteristics and sub characteristics. Figure 36 shows these quality characteristics.

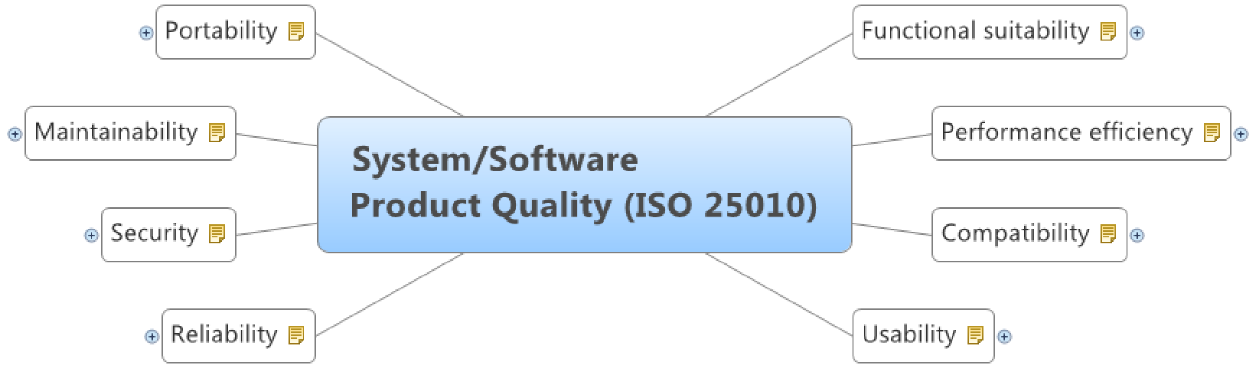


Figure 36: Product quality model and characteristics according to ISO/IEC 25010:2010.

Five sub characteristics are defined for "Security", ranging from confidentiality and integrity to non-repudiation, accountability and authenticity (cf. Figure 37). Additionally, the characteristic "Reliability" contains one sub characteristic which is usually associated with security, i.e. "Availability".

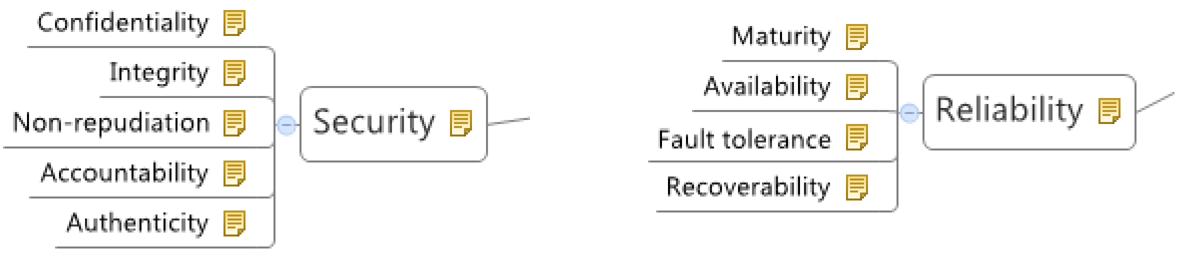


Figure 37: Sub characteristics for Security and Reliability according to ISO/IEC 25010:2010.

As it is imperative to understand the precise nature of each of these sub characteristics in the following discussion of the security ontology, their definitions are listed in the following Table 6.

Table 6: Definitions of sub characteristics of Security and Reliability.

Characteristic	**Definition (according to ISO/IEC 25010:2010)**
Security	
-Confidentiality	Degree to which a product or system ensures that data are accessible only to those authorized to have access
-Integrity	Degree to which a system, product or component prevents unauthorized access to, or modification of, computer programs or data
-Non-repudiation	Degree to which actions or events can be proven to have taken place, so that the events or actions cannot be repudiated later
-Accountability	Degree to which the actions of an entity can be traced uniquely to the entity
-Authenticity	Degree to which the identity of a subject or resource can be proved to be the one claimed
Reliability	
-Availability	Degree to which a system, product or component is operational

	and accessible when required for use

By combining the sub characteristics of Security and Reliability, the (ISO/IEC, 2010) quality model can be brought in line with the usual characterization of IT security predating its use. In this characterization, the notion of security was subdivided into three sub characteristics, called the C/I/A principle:

- Confidentiality,
- Integrity
- Availability

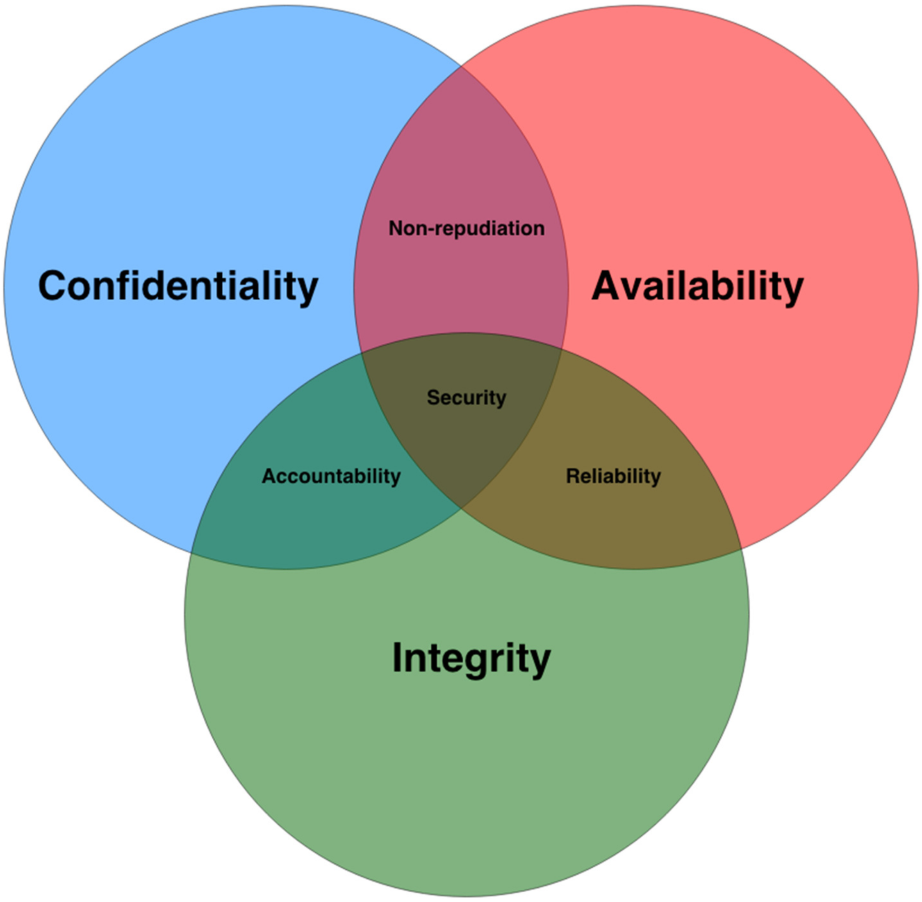


Figure 38: The Main Security Concepts as Defined in ISO 27000.

By using the presented sub characteristics, the set of sub characteristics now forms a true superset of the C/I/A principle sub characteristics as depicted in Figure 38.

4.1.2 Impact on Digital Preservation

The quality model relevant for security as presented above can serve as a typical example of how security is considered as part of requirements towards IT systems and software products and how security is modelled in quality assurance and testing to ensure that an IT system or software product really does contain the required aspects.

However, these requirements typically do not consider the needs of digital preservation; due to a lack of corresponding requirements, IT systems or software products do not contain provisions to easily enable digital preservation.

Additionally, security measures introduced into systems or products to satisfy security requirements such as the confidentiality or integrity of personal data or the authenticity of communication may actually prove to be great hindrances to successful preservation of a business process – either because they prevent preservation of key components, for instance because a software component prevents being copied or stored in a preservation repository or because they prevent redeployment, for instance because a software component requires an authentication server which may not be present anymore at the time of redeployment.

In a nutshell, digital preservation needs to put a special emphasis on security requirements and features of information technology components of business processes to be preserved in order to ensure the success of long-term preservation activities. Here, success can be characterised by the achievement of three objectives:

- **Preservation effectiveness**, i.e. ensuring that processes and systems can be successfully preserved and – at a later stage – redeployed.
- **Preservation efficiency**, i.e. ensuring that all preservation and redeployment activities can be executed in acceptable time and with acceptable effort despite the implementation of security measures in software components and IT systems.
- **Preservation fidelity**, i.e. ensuring that the preserved processes, systems and software components retain the required amount of original behaviour while at the same time documenting eventual fixity and provenance information.

In order to succeed, a preservation approach needs to identify the activities necessary to fulfil these objectives. As with software development projects, in digital preservation it is imperative to identify issues affecting the three objectives, and thus the outcome of the preservation as a whole, as early on as possible, due to the fact that issues identified late in the process may become economically infeasible or even impossible to correct.

4.2 Preserving Security

In general, the three objectives of effectiveness, efficiency and fidelity can be considered competing objectives, i.e. fulfilling any one objective may have a detrimental impact on any of the other two objectives. For instance, when trying to preserve a system that relies on an outside authentication service, the trade-off would be between ensuring preservation effectiveness (being able to preserve and redeploy at all), efficiency (not wanting to rewrite or purchase all rights to the authentication service just for the purpose of preservation) and fidelity (not wanting to replace the authentication service with a virtualized service stub that may not behave in exactly the same way as the original service).

In such a situation, preservation planning needs to provide guidance in order to choose the most appropriate trade-offs for any given case. To this end, prioritization between the three objectives depending on the concrete business process and its context needs to be made possible. Additionally, the necessary preservation and redeployment activities (such as setting up, testing and preserving a virtualized authentication service) need to be identified so that they can be undertaken as part of preservation and redeployment routine.

The TIMBUS preservation approach needs to provide this flexibility. In order to achieve this it builds on:

- the prioritization of security preservation objectives,
- a model of all relevant security-related aspects of the enterprise architecture,
- an approach of extracting these aspects, deriving preservation activities and storing them as part of provenance and fixity information,
- a process interface to include these activities into the TIMBUS process.

In this deliverable we want to develop a tool that enables the preservation of security knowledge that is relevant for a business process for the long term. In order to achieve this goal we need to define the scope and the vocabulary that are relevant for preserving the security information and the context in which is embedded.

From the TIMBUS perspective, the assessment of security requirements is key as the decisions that lead to the original implementation of security features is not within the focus of this deliverable. We require identifying the requirements and evaluating them against their impact on long term preservation and on redeployment. The selection of the appropriate service requires precise descriptions of the applied concepts and services. For preserving the knowledge about information security measures that have been used and implemented in a process, a formal model is required that describes the information in a precise manner. For this reason ontologies are an ideal candidate for mapping this knowledge as they provide the ability of making inferences on the status quo of a secured system.

Ontologies allow formalizing the knowledge of a certain domain by specifying the vocabulary to describe concepts, the relationships between concepts and the semantics associated with these relationships. The most cited[20] definition of ontologies within the context of computer science was given by (Gruber, 1995): [ontology is a specification of a conceptualization]. The specification is formal in way that it can be expressed with in an unambiguous way. This clearness allows translating the knowledge that is contained in such ontology into a computer interpretable format, such as RDF or OWL. As a result, ontologies can be used to describe arbitrary concepts of any domain in a machine understandable way.

The security ontology provides some of these aspects directly and supports the implementation of other aspects. The following sections will give an overview about related work, including other ontologies, before discussing the TIMBUS security ontology and its application as part of the TIMBUS process in detail. In this approach, the security ontology helps identify the relevant security context of the business processes to be preserved, helps to spot critical aspects that will lead conflicts between effectiveness, efficiency and fidelity and helps identify appropriate trade-offs as well as subsequent activities to ensure that all objectives are met to a sufficient degree.

[20] http://tomgruber.org/writing/ontology-definition-2007.htm

4.3 Related Work

With the increasing network linkage of our digital devices and the ever growing connectivity of the services offered, the need for information security has developed from a niche topic to a challenge topic relevant for a broad range of domains and organizations. The Commission of the European Union has expressed its urge for enabling and establishing a secure cyber infrastructure and will "work towards a coherent EU International cyberspace policy to increase engagement with key international partners and organisations, to mainstream cyber issues" (Commission, 2013). A precursory public consultation on "Improving network and information security in the EU"[21] showed that security incidents are increasing (57 percent of respondents had experienced security incidents in 2011 that had a serious impact on their activities) and require immediate actions for increasing the security of information systems. The European Commission responded with a "proposal for a directive of the European Parliament and the Council concerning measures to ensure a high common level of network and information security across the Union" (Commission, 2013).

4.3.1 Information Security

As our devices and services are highly interconnected, the needs for it protection from unauthorized access is a pressing issue. Although the awareness for the topic has increased significantly in recent years, there are still not full consensus about terminology and definitions (see (Bishop, 2003)). Computer security, cyber security, information security and other terms are often used in different contexts in academia and industry. To clarify and order the concepts and terms, this section provides an overview of existing work in the area of security standards, security ontologies and security taxonomies. The National Institute of Technologies' (NIST) Computer Security Division provides a glossary of key information security terms ((Kissel, 2013)) that allows settling the meaning for the relevant terms in this deliverable. In this deliverable, when we address computer security we follow this glossary and subsume all "measures and controls that ensure confidentiality, integrity, and availability of information system assets including hardware, software, firmware, and information being processed, stored, and communicated".

Many different standards exist that define security, its features and implications for complex systems. For preserving the knowledge that is required to run IT systems in a secure way, we need means for expressing the compliance of systems with security standards and their implementation details in a formal way. As it was already demonstrated in other deliverables, in the TIMBUS project we identified ontologies as a proper form of representing that knowledge and follow an approach that was already used in the construction of the context model.

Information security uses a different scope as it focuses on "the protection of information and information systems from unauthorized access, use, disclosure, disruption, modification, or destruction in order to provide confidentiality, integrity, and availability" (Kissel, 2013). Both definitions tightly entangled with

[21] http://ec.europa.eu/information_society/digital-agenda/actions/infosec-consultation/index_en.htm

each other as the protection of digital assets requires the implementation of the corresponding countermeasures.

The counter measures against potential and actual threats are called security controls. Implementing such defence mechanisms is always a cost driver (Cavusoglu, Mishra, & Raghunathan, 2004)and requires weighing the risk against the potential impact. Implementing security services and safeguards is not a trivial task and requires precise planning and effective implementation. For this reason several standards have been developed that define information security concepts and their implementation within the IT landscape and support the IT staff during the whole life cycle of the systems.

The most widely accepted standard in the area of information security is the ISO 27000 family of standards (ISO, 2012). It is depicted in Figure 39. The family includes 15 different norms that help organisations to run their IT infrastructure in a secure way. The different norms settle terms and definitions, describe how to gather requirements for secure systems, describe how to implement secure systems and measure their properties. Also the standard provides details about audits, risk management and governance. A glossary of terms can also be found in the freely available overview document of the ISO 27000 family. ISO 27001:2013, the international successor of the British Standard BS7799, specifies information security management systems (ISMS) and its relationship with other components within an organization. It describes the requirements for implementing a secure system and presents how these can be implemented, operated and maintained. ISO 27002 describes the security controls which are necessary for realizing the security requirements defined by ISO27001. ISO27005 introduces security risk management and describes how criteria for measuring risk and the assessment of these criteria can be evaluated (ISO, 2008). The following picture shows the components and the relationship between the different norms and standards of the ISO27000 family.

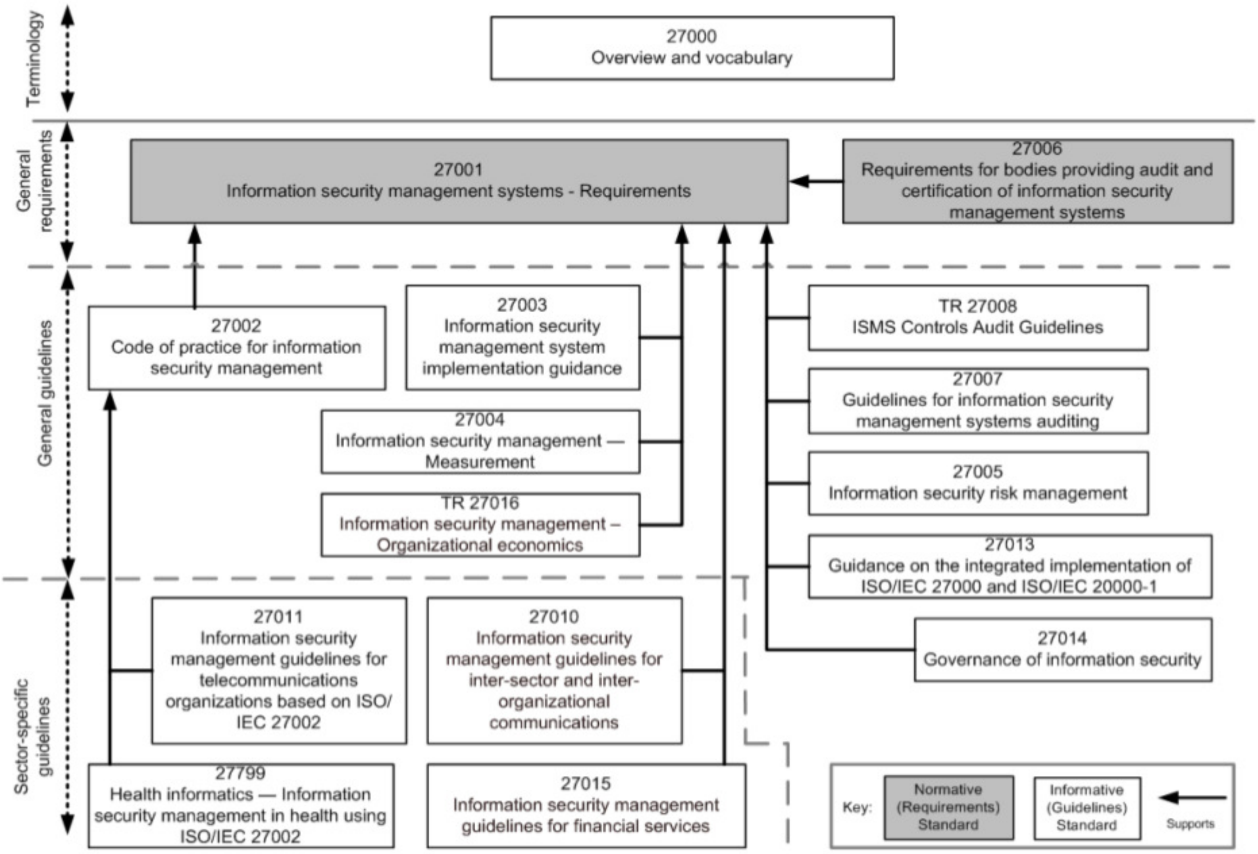

Terminology
27000
Overview and vocabulary
General requirements
27001
Information security management systems - Requirements
27006
Requirements for bodies providing audit and certification of information security management systems
General guidelines
27002
Code of practice for information security management
27003
Information security management system implementation guidance
TR 27008
ISMS Controls Audit Guidelines
27007
Guidelines for information security management systems auditing
27004
Information security management — Measurement
27005
Information security risk management
TR 27016
Information security management – Organizational economics
27013
Guidance on the integrated implementation of ISO/IEC 27000 and ISO/IEC 20000-1
Sector-specific guidelines
27011
Information security management guidelines for telecommunications organizations based on ISO/IEC 27002
27010
Information security management guidelines for inter-sector and inter-organizational communications
27014
Governance of information security
27799
Health informatics — Information security management in health using ISO/IEC 27002
27015
Information security management guidelines for financial services
Key:
Normative (Requirements) Standard
Informative (Guidelines) Standard
Supports

Figure 39: The ISO 27000 Family of Standards.

The aforementioned National Institute of Technology (NIST) offers a series of special publications on the topic of computer security[22]. The document SP 800-35 provides a generic guide for information technology security services and defines the six phases of the security lifecycle (Grance et al., 2003). These six phases are:

- Initiation: the decision of the implementation of a security mechanism
- Assessment: identify requirements
- Solution: select appropriate service
- Implementation: realization
- Operations: monitoring and periodic evaluation
- Closeout: transition to new service or service termination

These six phases can be recognized and integrated within our TIMBUS approach in order to describe the process of establishing, monitoring and replacing security relevant features during the lifecycle of the

[22] http://csrc.nist.gov/

preserved systems. How the assessment of security features can be performed is described in (Scarfone, Souppaya, Cody, & Orebaugh, 2008). From the risk assessment perspective the NIST document SP 800 – 30 provides a Guide for Conducting Risk Assessments (Blank & Gallagher, 2012).

4.3.2 Generic Security Ontologies

Ontologies describe and formalize knowledge about a specific domain which is necessary in order to define concepts and the relationships between them. An overview of ontologies in general and the TIMBUS ontologies can be found in (TIMBUS Consortium, 2013a). The aim of this deliverable is to provide a domain specific ontology (DSO) for security features of processes.

As security is a complex topic covering a complete profile of modern IT infrastructure, ontologies have been recognized as important and useful tool for describing this knowledge precisely (Donner, 2003). To achieve this goal, a common language is needed that consists of at least two parts: high level terms and taxonomy (Raskin et al., 2001). They allow agreeing upon a certain vocabulary and restricting the domain. In addition, ontologies allow describing complex settings in a manner that human beings can get an overview of a specific domain with a graphical representation, but also machines can process the contained information automatically.

Several different ontologies have been proposed, providing different viewpoints, focus and complexity. A generic ontology of information security was proposed by (Herzog, Shahmehri, & Duma, 2007). The ontology uses a total of more than 300 classes for describing assets, threats, vulnerabilities and countermeasures, which are set into relation with 34 properties (relations). Picture Figure 40 shows an overview of the ontology.

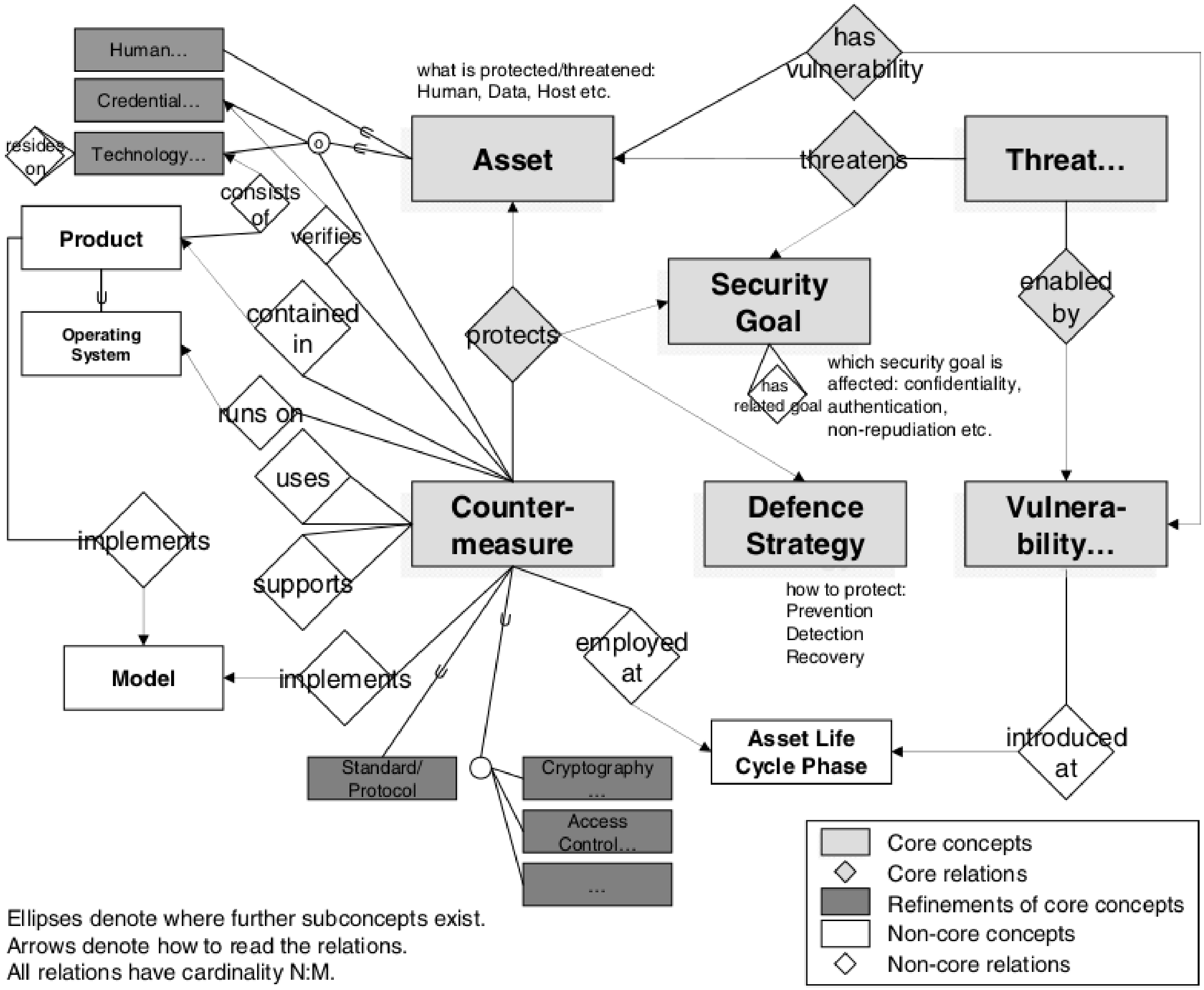


Figure 40: The Generic Security Ontology by (Herzog, Shahmehri, & Duma, 2007).

A general purpose ontology was presented in (Denker, Kagal, Finin, Paolucci, & Sycara, 2003). The ontology is used for annotating Web resources and provides information about their security features. The core concepts are security mechanisms, notations, signatures, protocols, key formats, encryption and syntax. Picture Figure 41 shows an overview of this ontology.

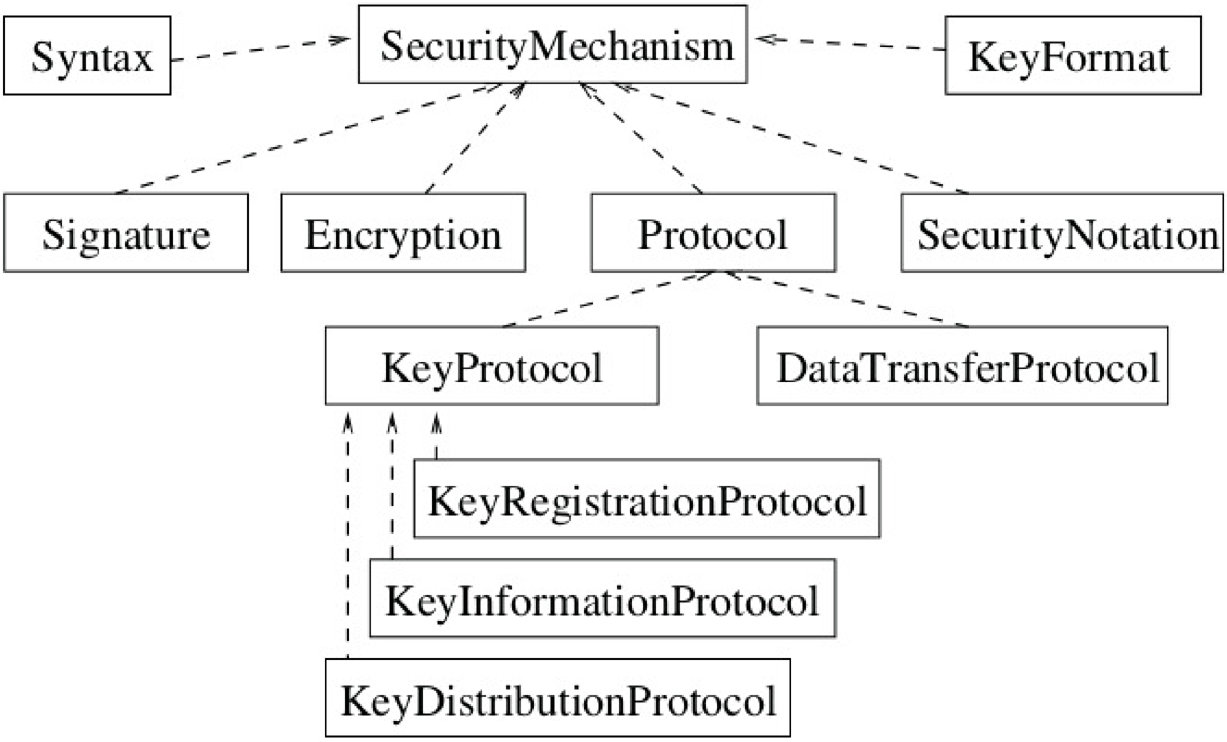


Figure 41: The Security Ontology from (Denker, et al., 2003).

Based upon the basic ontology for annotating Web resources, the Naval Research Laboratory (NRL) developed a more complex ontology following a modular approach (Kim, Luo, & Kang, 2005). The aim of this ontology was not describe security concepts not exclusively for Web services, but for general resources. Another aim was to maintain the ontology extensible and provide various levels of details suitable for different purposes. The ontology consists of seven separate sub ontologies covering the areas algorithms, assurance, service security, agent security and information objects and a central main ontology containing general terms. This main ontology specifies the three main classes SecurityProtocol, SecurityMechanism and SecurityPolicy which are subsumed under the top class SecurityConcept. Picture Figure 42 shows this main ontology.

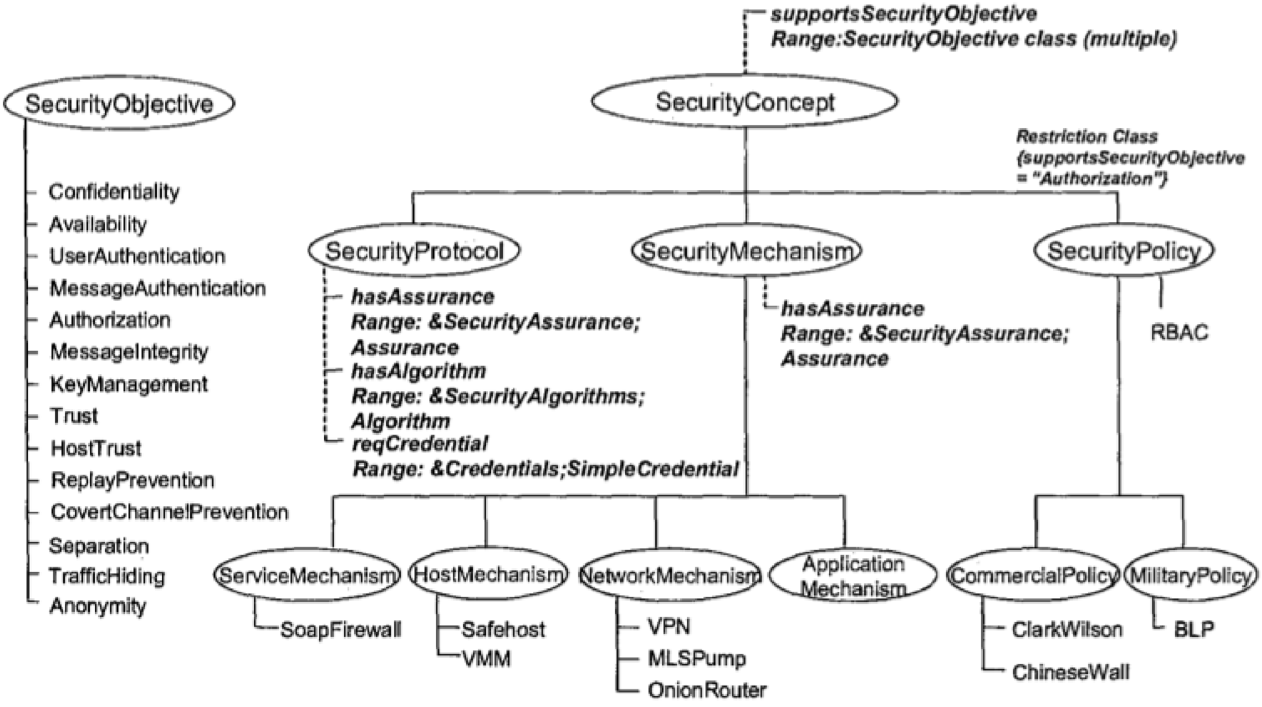


Figure 42: The NIST Security Ontology by (Kim, Luo, & Kang, 2005).

Wherever possible the NRL ontology aims to map concepts to existing NIST standards by using a NIST Standard property.

One of the most extensive information security ontologies was proposed by Fenz et al. (Fenz & Ekelhart, 2009). The ontology can be mapped upon the ISO 27001 standard and therefore provides all concepts introduced in the ISO standard. Furthermore it incorporates the German IT Grundschutz[23] Manual (see (Münch, 2008)) in order to achieve an even broader coverage of well-established security standards. The ontology uses 500 concepts and more than 600 formal restrictions in order to describe the domain. It is separated three sub ontologies covering security, enterprise and location concepts. The relations between the classes are considerably precise and formal axioms model aspects from the physical environment as well as details of the security controls. Figure 43 shows a high level overview of the ontology.

[23] https://www.bsi.bund.de/EN/Publications/BSIStandards/standards.html

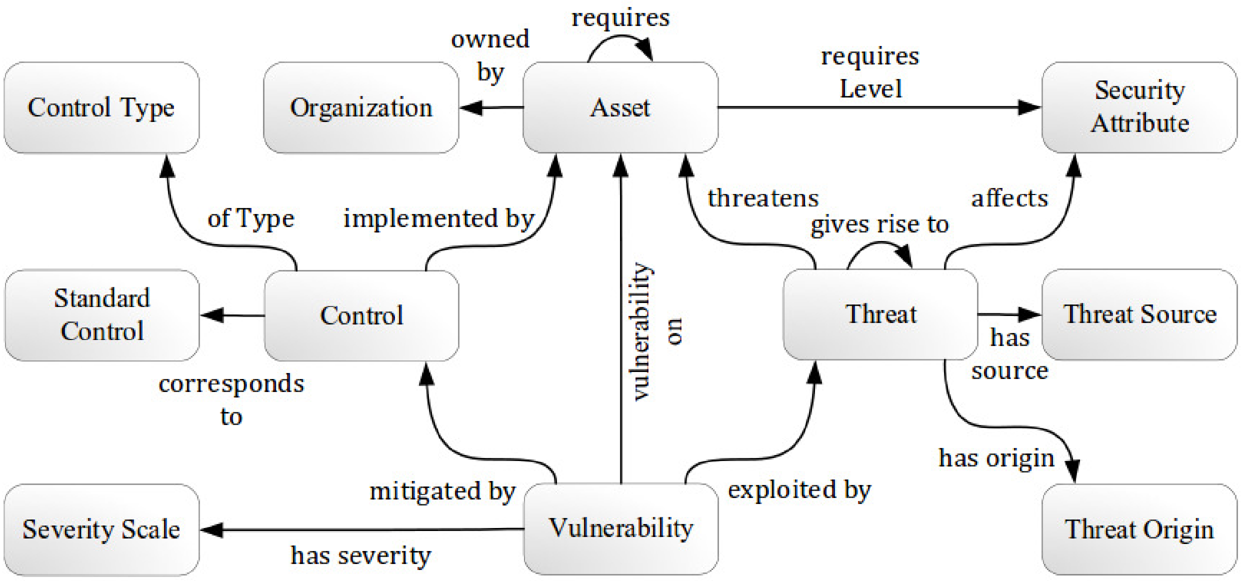


Figure 43 The Security Ontology by (Fenz & Ekelhart, Formalizing information security knowledge, 2009).

The authors extended their ontology (Fenz & Ekelhart, 2009) by introducing metrics that allow evaluating the quality and compliance of an enterprise and its security infrastructure according to the ISO 27001 standard. Thus companies can assess their information security infrastructure and policies continuously throughout the security life cycle.

4.3.3 Security Requirement and Risk Ontologies

In (Tsoumas, Dritsas, & Gritzalis, 2005) a framework is presented that shows how security ontologies can be used in order to associate the security requirements that have been identified with their actual implementation. This relationships can be queried and allow to derive new knowledge about the security features of a system. The authors identified several sources for that knowledge. This includes high level security information, best practices and standards or technical infrastructure details, vendor information or data available from public security portals.

The authors of (Lasheras, Valencia-Garcia, Fernandez-Breis, & Toval, 2009) provide a model for describing security requirements with ontologies. The framework incorporates existing security and risk management standards and allows reusing existing knowledge about the security requirements and sharing this knowledge.

A further application of security ontologies covers the area of risk management within the domain of information security. Ontologies as proposed from Ekelhart et al. (Fenz, Ekelhart, & Neubauer, 2009) allow enterprises to assess the risk of their infrastructure and use the framework for taking decisions regarding security investments.

4.3.4 Taxonomies

Taxonomies are hierarchically structured classifications of terms of a given domain of interest. There exist several taxonomies that arrange and order the terms, definitions and concepts used in the area of information security. In contrast to ontologies, taxonomies are strictly hierarchical and do not allow arbitrary relationships between classes. The similarities and differences between taxonomies and ontologies are described in the appendix of (Kim, Luo, & Kang, 2005).

In (Savolainen, Niemela, & Savola, 2007) the authors introduce five main categories: security assets, security attributes, SecurityThreats, SecuritySolutions and SecurityMetrics. Picture Figure 44 depicts the taxonomy.

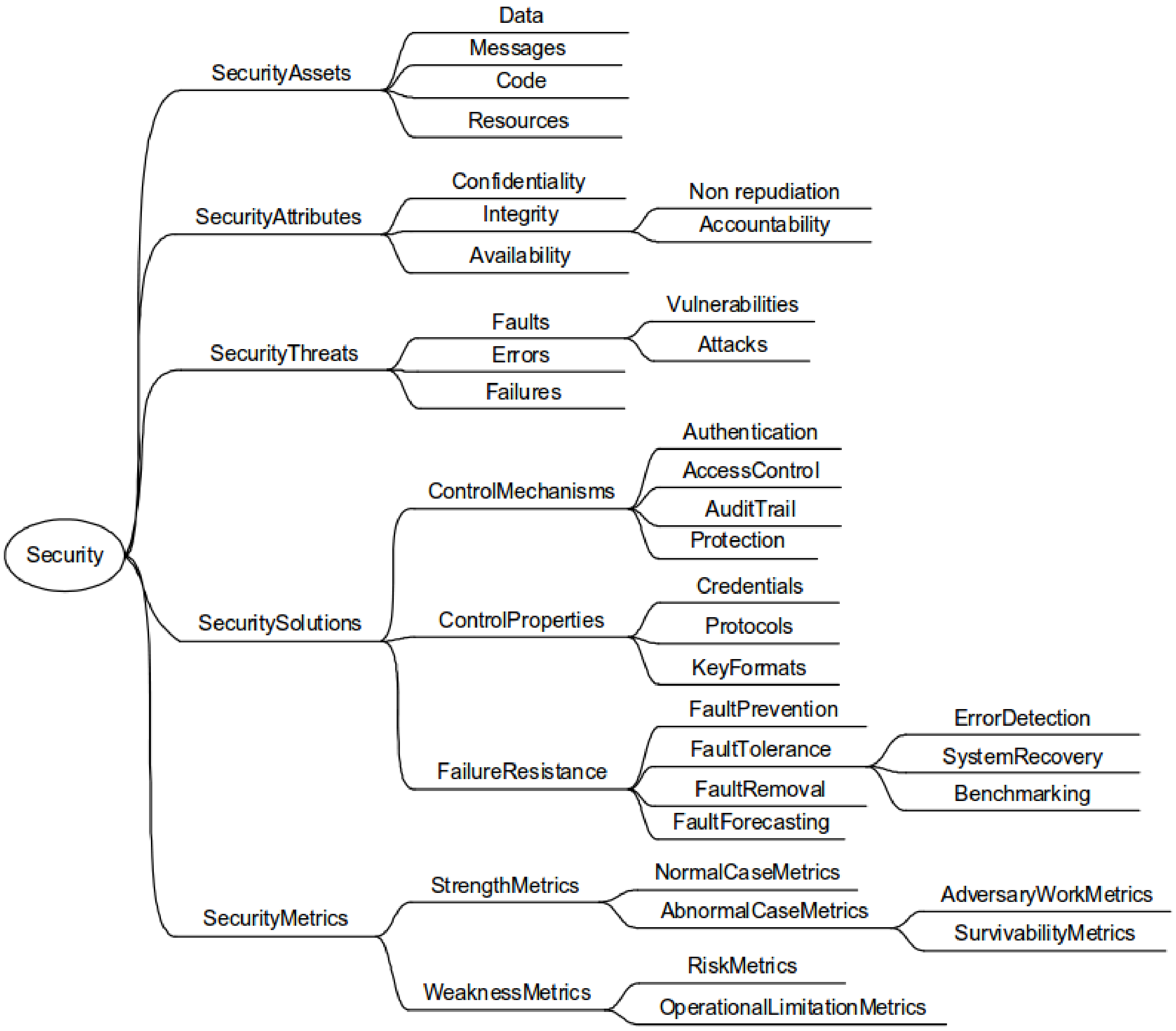


Figure 44: The Taxonomy of (Savolainen, et al., 2007).

A generic taxonomy of security related requirements is described in (Firesmith, 2005). The author identifies different security requirements that are derived from existing safety taxonomies. Although safety and security are closely related terms, these should not be used interchangeably. Safety deals with accidental harm whereas security treats malicious harm. The taxonomy used four basic classes: pure security requirements, security-significant requirements, security system requirements and security constraints. The class pure security requirements cover 16 subtypes and define minimum level security levels for these. Security-significant requirements incorporate risk analysis based of different subtypes such as threats and incidents and arrange them into their relative security risk. The security systems class covers architectural properties and constraints deal with engineering decisions.

The authors of (Venter & Eloff, 2003) provide a taxonomy of information security technologies and describe in detail how this hierarchical representation was established. A taxonomy of security metrics was proposed in (Savola, 2007). The taxonomy is divided into several layers; the root node covers business layer security metrics such as trust, risk, information security, cost and dependability. These layers are then subsequently refined in order to express the subclasses of the appropriate main category.

A further taxonomy from (Avizienis A. , Laprie, Randell, & Landwehr, 2004) uses the concept of dependability, which combines general terms such as reliability, safety, integrity to a more global concept. The paper describes the process of extending taxonomy with security features that have been introduced with the arrival of new technologies. The taxonomy also has a focus on threat scenarios and provides a sub taxonomy describing failures and errors that can occur. An overview of this taxonomy is depicted in the Figure 45.

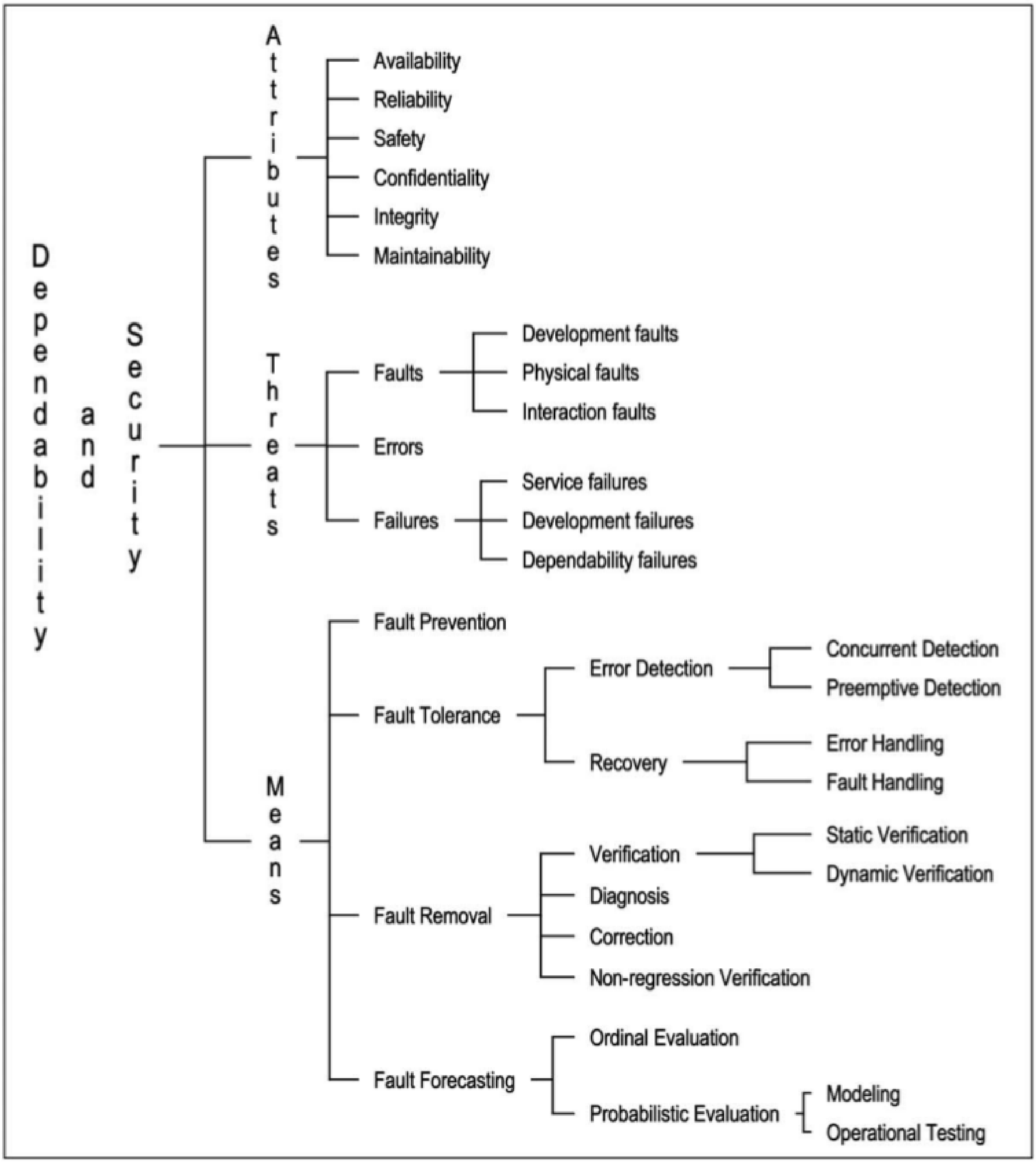


Figure 45: The Security Taxonomy of (Avizienis, et al., 2004).

Based on the Taxonomy of Avizienis et al, a ontology was developed by Fenz et al. that allows to quantify the risks that threaten to an organization (Fenz, Weippl, Klemen, & Ekelhart, 2007). This so called heavy weight ontology allows to assess the security of a certain setup with respect to persons and to physical properties as well and relate the risk to costs. Heavy weight ontologies incorporate axioms and constraints on entities, which include more semantics than lightweight descriptions of hierarchies and relationships can provide.

4.3.5 Literature Surveys and Requirements Engineering

In (Souag, Salinesi, & Comyn-Wattiau, 2012) the authors provide a classification of security ontologies into eight families. They distinguish security ontologies, security taxonomies, general security ontologies, specific security ontologies, Web oriented security ontologies, risk based security ontologies, security ontologies focused on requirements and meta models for security ontologies. The work provides an evaluation of the mentioned security ontologies and shows whether the ontologies can be used for requirements analysis. As expected the generic taxonomies such as (Fenz & Ekelhart, 2009) or (Kim, Luo, & Kang, 2005) cover the most security related concepts which are relevant for our work.

The authors of (Blanco C. , Lasheras, Valencia-Garcia, Fernandez-Medina, Toval, & Piattini, 2008) provide a systematic review of existing security ontologies and compare them according to criteria such as number of classes, instances, and depth of hierarchies or average relations between the classes. Later, the authors repeated their survey in (2011) and extended the coverage of security ontologies they considered. The authors conclude that an integrated ontology that includes all the knowledge of the field of information security is still hard to achieve because the field is highly diverse and complex.

Although there exists a variety of security related ontologies, none of the presented works could be used in order to describe the security features of business processes in the appropriate level of detail. The existing ontologies introduced in Section 4.3 either have the focus of describing potential threats, attack scenarios or aim to derive potential vulnerabilities from the described security assets. The TIMBUS Security Ontology has a different focus. The aim of this ontology is to describe security concepts and their implementation within a business process or a specific subcomponent. Similar to existing ontologies such as (Fenz & Ekelhart, 2009), it links assets to security controls.

We decided to avoid the overhead of either extending or reducing existing ontologies until they fit into the TIMBUS processes and design our own TIMBUS security ontology, which is lightweight and still expressible enough for describing security features of business processes and their IT landscape in a level that allows later redeployment.

4.4 Solution Overview

Protecting information infrastructures and applications from unintended usage is becoming more and more important as there is a tendency of orchestrating different services and distributed components to achieve a business goal. Services need to transmit sensitive data over insecure channels, provide multiuser access with different roles and permissions, encrypt local files and prevent complete systems from unauthorised access. Information security is a domain utilizing a large amount of different concepts ranging from generic principles such as confidentiality, authenticity or integrity to highly implementation specific concepts. We require methods that can describe the security features regardless of their scope and preserve the knowledge about the security features for the long term. In order to do so, the vocabulary needs to be specified in a precise way. An ontology does not only define the used vocabulary and formalises the relationships between the concepts explicitly. It also allows to be filled with instances of the concepts. Utilizing the real world objects and the knowledge represented by the ontology and its relationships allows deriving answers to the questions about the domain. Therefore the knowledge can be shared and reused in different scenarios and existing ontologies can be combined with other knowledge representations. This enables the combination of various domains and describes complex areas of interest.

The goal of the TIMBUS Security Ontology is to provide knowledge of basic security concepts and store this information for the long term. We designed the ontology in a generic way because the ontology should allow the mapping towards other ontologies that are specialized in a domain. This enables domain experts to integrate the knowledge of the security details of a business process into an appropriate security ontology that might not even exist at the point of writing this deliverable.

4.4.1 Applications for a Security Ontology

Our approach follows an abstract model which allows us to describe the use cases and their security features in a level of detail that is tailored to fit into the existing context model that is used for describing processes in TIMBUS.

The perspective on security that we implemented is oriented towards the three phases of the TIMBUS life cycle. The ontology needs to answer the following questions with respect to planning, preservation and redeployment phases:

- Who?
- What?
- Where?
- When?
- Why?
- How?

Within the security context these questions refer to actors that have certain privileges on specific systems during a defined time period in order to fulfil a purpose. Actors can be users that authenticate against a system and then get are equipped with a specific set of permissions. Actors can also be software or other processes that require access to a resource. The ontology needs to describe what kinds of privileges are defined and how they are used within the context of a process. It needs to specify how the permissions are granted and allow modelling the level of details that is required, i.e. the ontology needs to describe if a business process is secured by a general principle in the same manner as a specific database table can require precisely defined access constraints. Furthermore details about the implementation of security controls need to be mapped as well as their properties and technical features.

Therefore the aim of the TIMBUS ontology is to describe the security features that have been implemented by a process and associate these with users, abstract roles, files, services or sub processes during all three phases of the TIMBUS life cycle. This description can be used for protecting sensitive data for the long term and for managing and maintaining data with an appropriate level of security.

This collection of security knowledge associated with a business process serves as an inventory of the security features. Whenever a certain technology gets obsolete, an algorithm broken and a security standard revised, the ontology can be used for finding all critical implementations and replace the security control in question with a current version in order to restore a secured version of a preserved business process.

Digital preservation aims to keep information and knowledge accessible and to maintain this availability for future generations. Hence all methods that limit access to data and information are a potential threat to digital preservation and therefore to the goals of TIMBUS. Information security methods such as encryption add an additional layer of complexity to the already challenging problem of preservation.

As every other software, encryption libraries can become obsolete quite quickly and newer versions might not guarantee backward compatibility. Also used encryption algorithms might get insecure and hence lose their purpose. The same is true for authorisation mechanisms and permission systems. Preserving complex systems and sensitive data is a complex task. There is always a trade-off between the complexity of the preservation actions and the level of security that has to be maintained. Therefore TIMBUS also needs tools for removing additional levels of security where they are not needed. This includes for instance the abstraction of individual access roles from actual users into generic roles with reduced complexity. Also methods that allow the removal of encryption or the replacement of potentially complex authentication and authorization with simpler yet sufficiently secure mechanisms need to be supported.

This diverse set of requirements demands a flexible solution that is independent from actual implementations today, but able to express security requirements that fit into future scenarios of secure business processes.

4.5 The TIMBUS Security Ontology

The goal of the TIMBUS Security Ontology is to describe all security aspects that have an impact on any of the three phases of TIMBUS and preserve the knowledge of these aspects for the long term. To achieve this goal, the ontology has to meet several requirements.

The main requirement is the coverage of all security relevant features that are used in our use cases and a description that is precise enough to define how a certain aspect of a business process requires protection and how this is implemented. A further requirement is the ease of use, i.e. the ontology needs to be generic enough to allow non-security ontology experts to identify key concepts. Also the ontology needs to be sufficiently flexible in order to address high level concept and specific details.

4.5.1 Designing the Ontology

Based on the previous security ontologies that have been introduced in Section 4.3 we built an ontology that can be integrated into the TIMBUS Context model seamlessly. In order to analyse which core concepts are required, we followed the used cases in a bottom up approach and refined the ontology using top-down principles.

We interviewed the use case providers in order to identify the relevant security aspects. An example for such a questionnaire can be found in Appendix B. This allows us to maintain a list of concepts that our ontology needs to consider.

In the second step, we created a visual representation of the concepts (see Figure 46) that have been identified either via questionnaires or by the direct analysis of the use cases. We mapped the components to their corresponding entities based on their Archimate Models and associated them with security concepts.

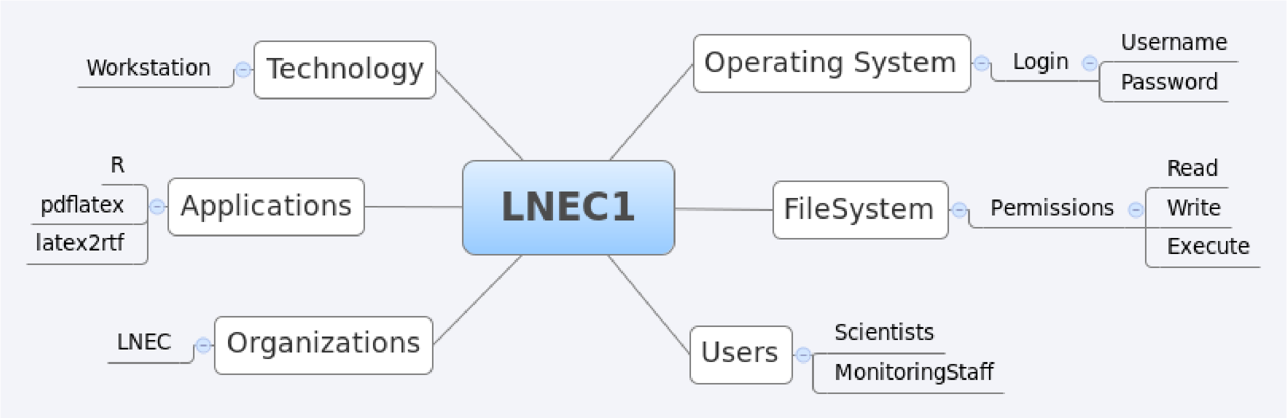


Figure 46: A Simplified Overview of the LNEC1 Use Case.

Based on the results we retrieved from the use case owner, we formulated questions that the ontology has to answer. Typical questions are:

- Is the data encrypted?
- How do users authenticate?

- What privileges does a user need to execute a process?
- Can different users run the process?
- Was the data unchanged?
- What permissions does an authenticated user need to execute a process?
- What skills does the administrator need?
- Do we have to encrypt all data?
- Is there a policy in place?
- Can we process emails?
- Who approves and allows access?
- How are permissions stored?
- Until when is a password valid?

In a following step, we evaluated the concepts that are required for answering these questions in a top down fashion. Not to reinvent the wheel we analysed the coverage and the applicability of existing approaches. We realized that some of the ontologies are too specific to be used by non-domain experts who need to describe their service infrastructure. Therefore we required a more generic approach to include concepts that describe high level security requirements.

In order to understand which concepts are commonly used, we analysed the literature of a professional information expertise certification program (Harris, 2012). The Certified Information Systems Security Professional (CISSP) programme is a recognized and established information security certification that is developed by the International Information Systems Security Certification Consortium[24]. We extracted the concepts and included them into our TIMBUS Security Ontology. The rationale behind this approach was to integrate the knowledge about information security that is available to a certified professional. Also the structure that was used in the CISSP literature allowed us to align the topical clusters in a similar way, thus the usability of the ontology was increased.

4.5.2 The Ontology in Detail

The ontology consists of 156 classes, 29 object properties and 5 data properties. The classes are hierarchically organized under subtopics. These topics are listed in the following table and described in more detail below.

[24] https://www.isc2.org/

Table 7: The Main Topics Covered in the TIMBUS Security Ontology.

Topic	Short description
Artifact	Any (digital) object that might have a security control associated with it.
AuthenticationMechanism	Authentication mechanisms and their implementation
AuthorizationMechanism	Authorization mechanisms and their implementations
Concepts	Main concepts from the security domain, e.g. authentication, authorization, availability, confidentiality etc.
HumanResources	Competencies, education and responsibilities
Permissions	Different permission modes
Users	User roles

4.5.2.1 Artifact

In the context of the TIMBUS Security Ontology we define artifacts as all those things which can have an associated security related context. Other security related ontologies use the term asset to denote data, devices and information that requires protection. We chose the term artifact as it is used more often in the digital preservation community where we see the main applications of our work.

In this context we denote all digital objects that are generated, produced or used either by humans or automated agents during the execution of a process as artifacts as well. We include processes in this definition as well as they can have associated security permissions and require their own abstract regulation. Therefore, artifacts include all documents, spreadsheets, configuration files, diagrams, databases and database objects, source code, metadata or even complete processes and services.

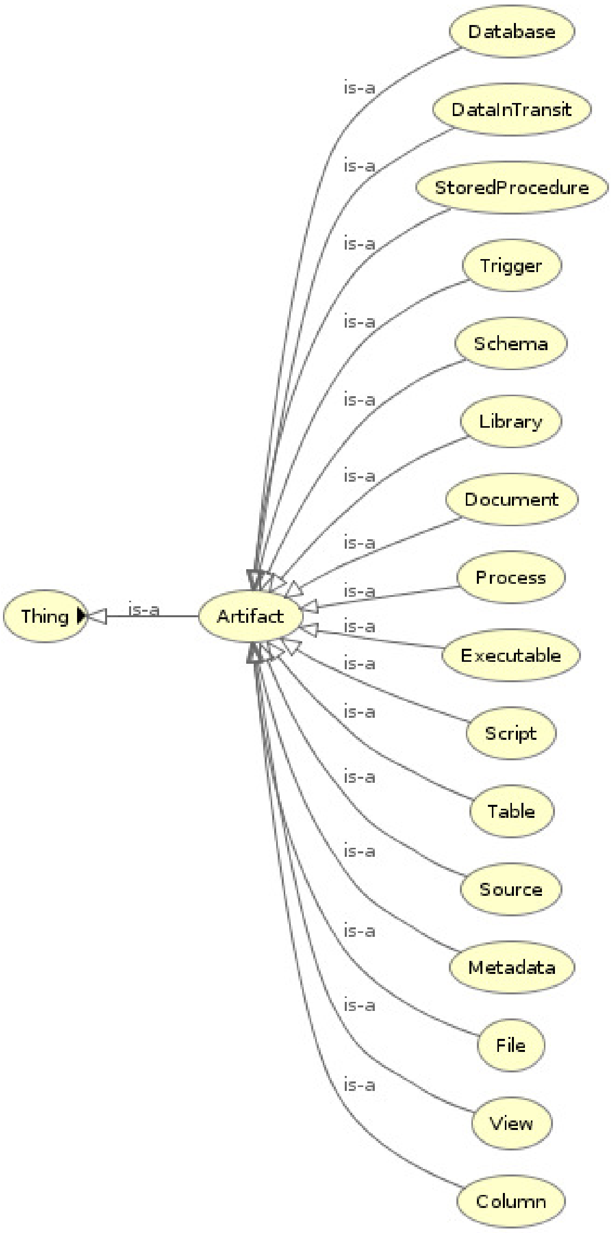


Figure 47: The Artifact Classes.

This sub-branch is used in order to associate artifacts with their security requirements or features in a fine grained manner. These classes can be referred to by the Archimate models that have been developed for the use cases in order to express security concerns for a process. Discussed concepts are presented in Figure 47.

4.5.2.2 Authentication Mechanisms

In order to enforce security constraints and regulate access to artifacts, several aspects have to be considered. Business processes differentiate between different roles that have diverse sets of permissions, competencies, rights and rules associated. These different roles are implemented and realized by IT systems which enforce the compliance of users with these rules. With users we subsume human actors, software agents and processes.

In order to accomplish the rules defined for certain roles, IT systems require ensuring that the interacting actor is actually who he claims to be and then only grant those permissions which are provided for this role while dismissing all other activities. The sequence of steps is depicted in Figure 48.

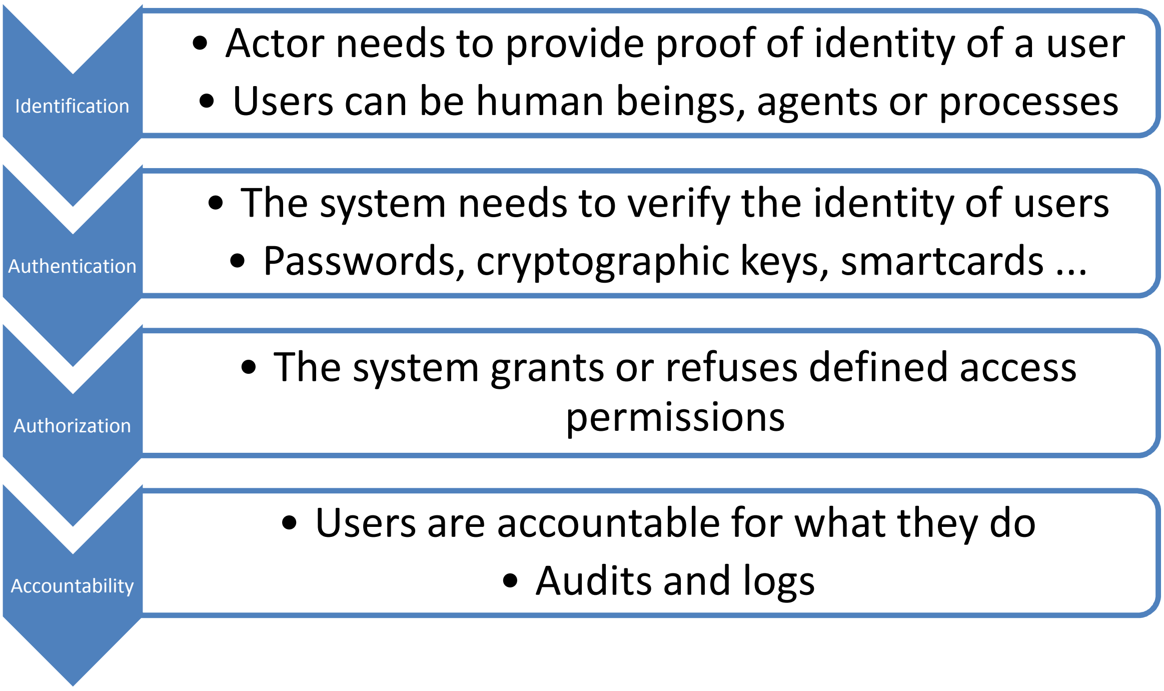

Identification
Actor needs to provide proof of identity of a user
Users can be human beings, agents or processes
Authentication
The system needs to verify the identity of users
Passwords, cryptographic keys, smartcards ...
Authorization
The system grants or refuses defined access permissions
Accountability
Users are accountable for what they do
Audits and logs

Figure 48: Authentication and Authorization Steps.

Authentication denotes the process of ensuring that two interacting parties are actually who the purporting to be. Authentication does not only refer to human beings proving their identity, but also to software systems which need to provide proof for their identity. Several authentication mechanisms have been implemented, which are listed on the sub-ontology shown in Figure 49.

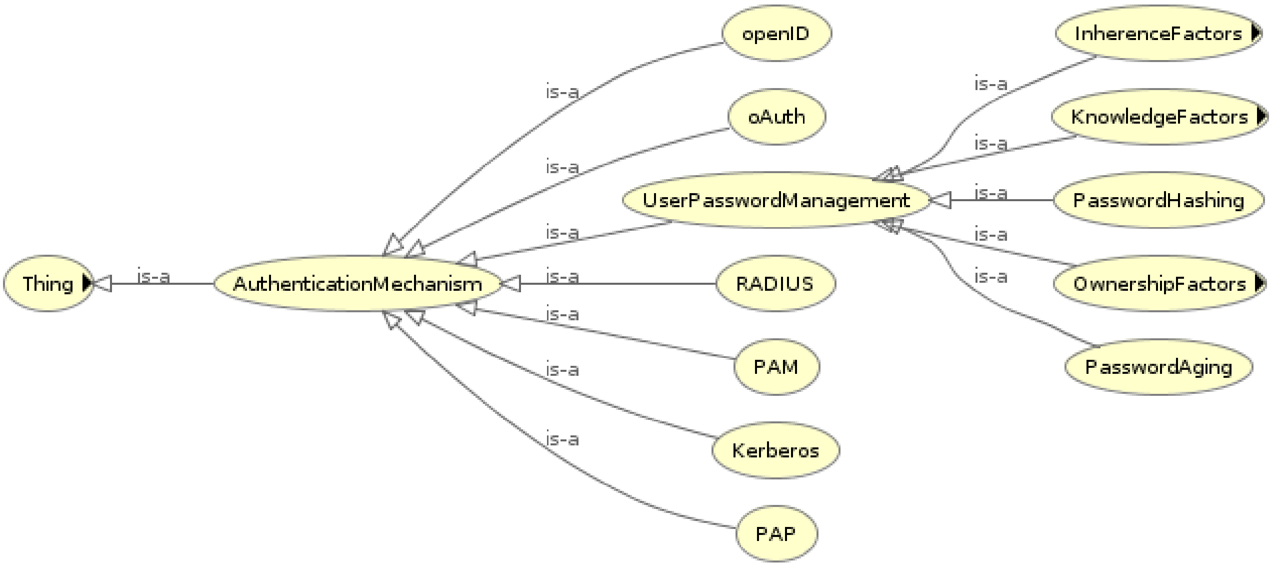


Figure 49: Authentication Mechanisms.

Generally speaking there are three factors that can be used for authenticating users with a system. According to the authors of (Harris, 2012) we included the following factors: something a user knows (e.g. passwords), something a user has (e.g. key cards) or something a user is (e.g. biometric features). It is clear that the information that is used in order to authenticate a user with the system (e.g. a password) needs to be preserved a long with the process. Figure 50 shows these factors and their representation in our ontology.

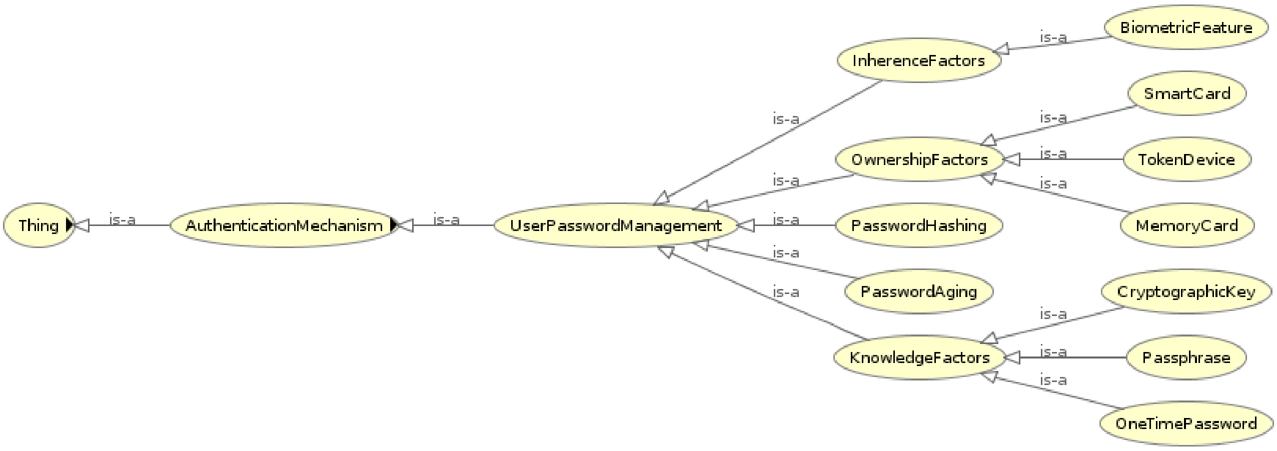


Figure 50 Authentication Mechanisms.

The ontology we provided either allows attaching the corresponding item to the model as an instance and preserving the information together with the process metadata. In cases where this is hardly possible (e.g. fingerprints), the ontology provides guidance how an authentication mechanism can be replaced with a proper format for long term preservation.

4.5.2.3 Authorization Mechanisms

Authorization is the process of granting permissions which enable an agent to perform a certain action. This process is closely related to authentication, but may not be confused as the two are fundamentally different processes. Authorization is responsible for checking if an agent has sufficient rights to access a specific object. It is applied after a user has been authenticated. Authorization mechanisms describe in detail the different permission levels and enforce them by the use of appropriate controls (see Figure 51).

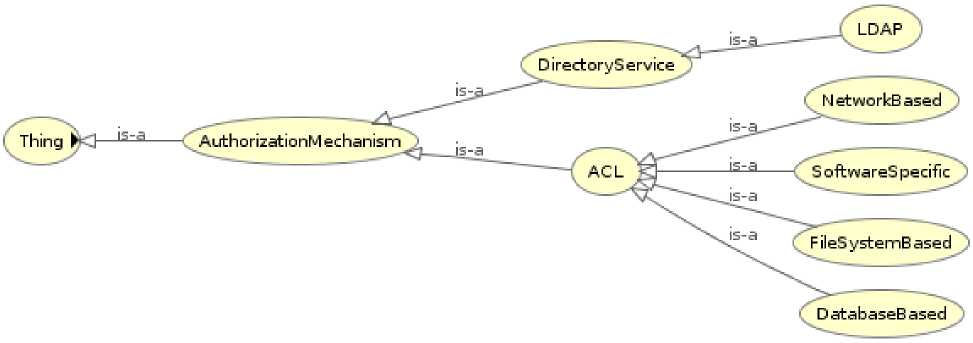


Figure 51: Authorization Mechanisms.

The most common systems use access control lists (ACL) or directory services such as LDAP. These systems contain valuable information how each role was designed for a specific purpose within the process and which permissions have been associated with it. This information is required when actual users (e.g. employees) and their permissions need to be mapped to abstract user role in order to enable long term preservation.

4.5.2.4 Permission Classes

In order to model fine grained privileges and constraints which describe how and by whom a specific artifact can be accessed, modified or deleted, a set of permissions is required. On the one hand the permissions describe what set of privileges is granted for a specific role. On the other hand the permissions are associated with specific artifacts, e.g. database tables or views. Figure 52 shows the permissions.

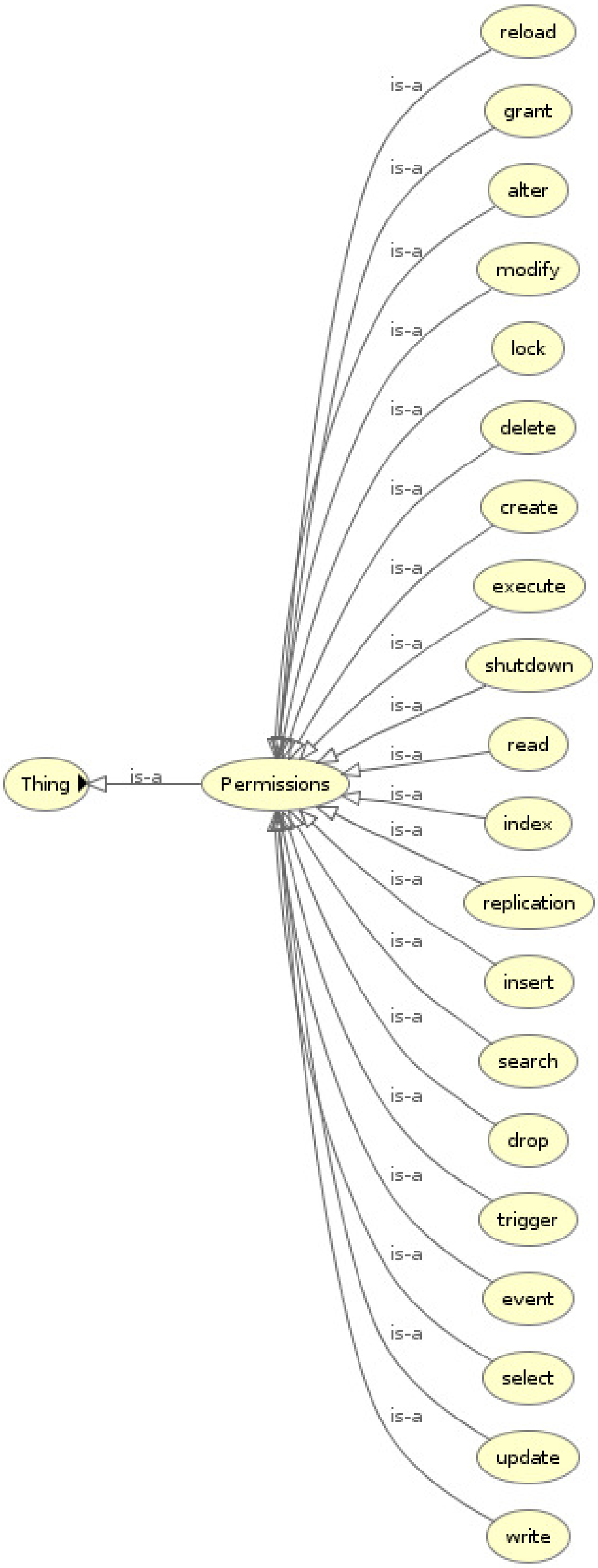


Figure 52: The Permission Classes.

4.5.2.5 User Roles

One goal of the TIMBUS Security DSO is to describe user roles in an abstract way. The following shows the user roles that have been modelled according to the proposed classes of users in (Harris, 2012).

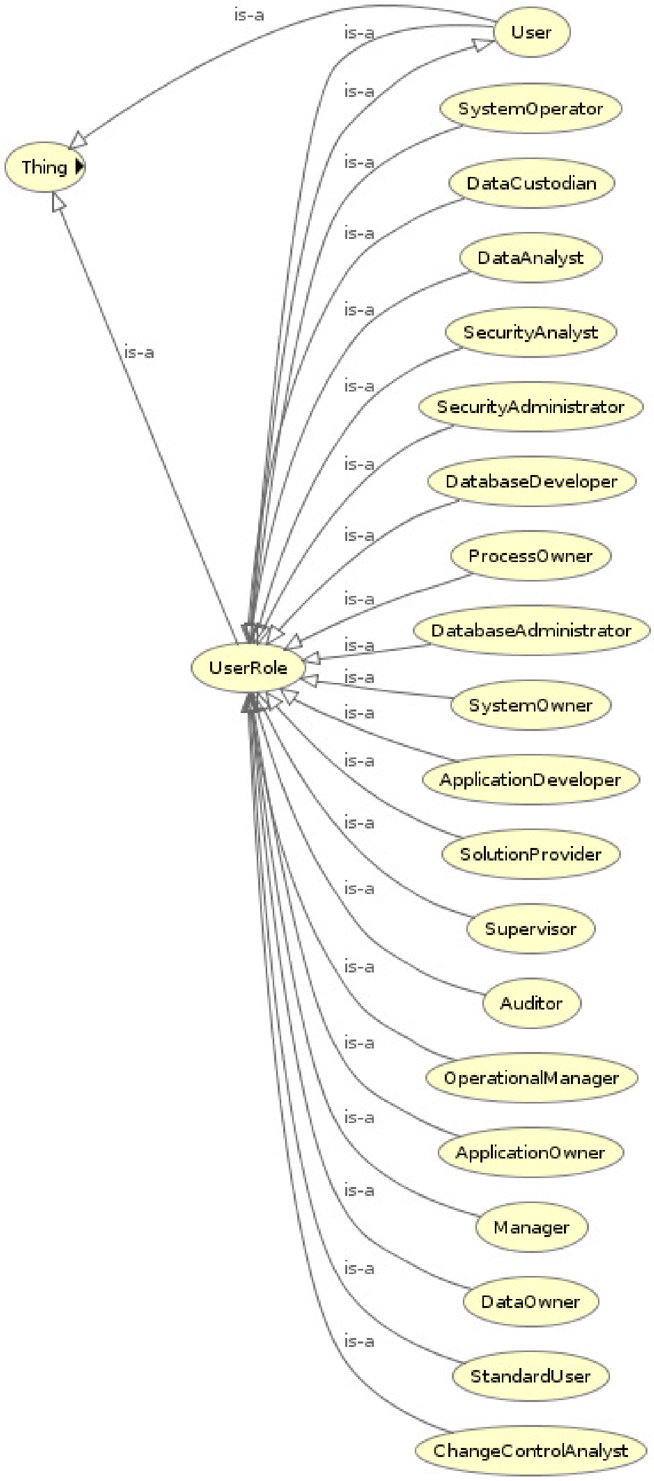


Figure 53: User Roles.

The user roles can be equipped with the appropriate permission sets as introduced in Section 4.5.2.4 and also be mapped with human resources concepts which are provided in Section 4.5.2.1.

4.5.2.1 Human Resource Concepts

Furthermore we provide basic classes for describing competencies, skills, required trainings and other human resource related concepts. These can be used in order to associate a specific role with the skill set that is required by any user of a role. Figure 54 shows these concepts.

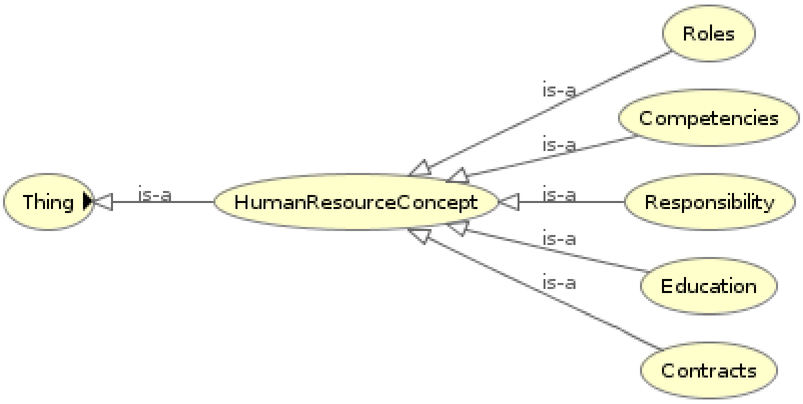


Figure 54: The Human Resources Concepts.

These classes can also be used in order to associate documents such as contracts or regulations with a specific role and thereby integrate additional metadata directly.

4.5.2.2 Concepts

As described in Section 4.1.1, information security pursues several security objectives. These objectives are modelled as concepts within the ontology. Information security is built on top of these concepts that each enables a different core principle of secure systems. The ISO 27000 family of standards defines information security as the preservation of confidentiality, integrity and availability[25]. The standard also recognizes authenticity, accountability, non-repudiation and reliability.

Only if these security properties are considered, an information system can be considered secure. The sub ontology depicted in Figure 55 shows the concepts that we considered essential for being preserved.

[25] https://www.iso.org/obp/ui/#iso:std:iso-iec:27000:ed-1:v1:en

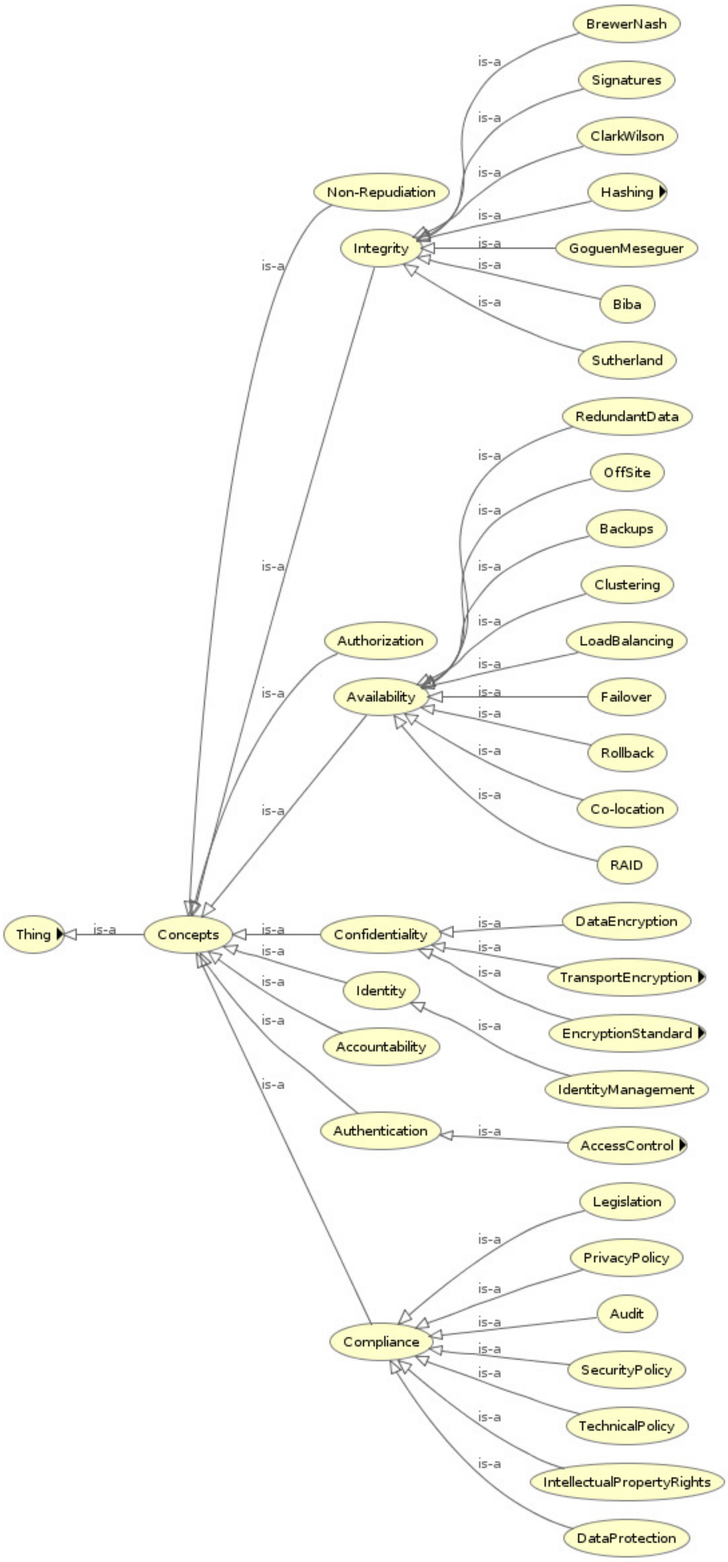


Figure 55: Security Concepts.

Non-repudiation is a concept that ensures that every properly authenticated user can be made accountable for the actions he performed in a system. No user that was involved in a process can deny at a later point that he or she was involved in the execution of any operation. Non-repudiation is established by collecting evidence about all actions that a user performed in a way that it is provably ensured that to false claims can be made (Zhou & Gollmann, 1997). The information gathered constitutes to the provenance track.

Integrity is a principle that ensures that data is not altered in any unauthorized way. This includes that data remains complete, correct and accurate. Every alteration of data must be recognized, either if the changes were triggered by an intentional user or by an unintentional software bug. Usually so called cryptographic hash functions are used to compute individual checksums for each file. These checksums can then be used in order to detect any changes within a file (Paar & Pelzl, 2010). The ontology allows mapping different hash functions and concepts of integrity schemes.

Availability refers to the concept that data and information must be available to users at any time they request. Availability can be achieved by several approaches such as failover systems and backups. In the context of preserving security information the concept of availability can be used in order to denote systems that are not within the scope of the current process, but where sensitive data was stored for instance as a backup.

Confidentiality is essential when dealing with sensitive data that needs to be protected from unauthorized access. The concept of confidentiality ensures that only agents having the correct permissions, credentials and access rights can read or modify data. Confidentiality can be achieved by encrypting data. We differentiate between data in transit and local data. Data that needs to be transferred via potentially unsecure channels requires different encryption standards to be in place in comparison with files that need to be encrypted during the whole preservation process. The ontology we provide can be used in order to determine which encryption algorithms have been used in both cases. Thus for data that is intended to be protected only during its transmission, it is sufficient to replace the encryption and decryption modules with never implementation that are still secure in a new environment. The ontology allows finding these algorithms and retrieving their descriptions as they are available in the confidentiality branch of the ontology. Data in transit and data that need to be archived in a secure way differ also in the requirements in terms of robustness over time. The verification process may require transmission data for verification purposes. Hence it needs to be stored and mapped to appropriate descriptions as well. Data that needs to be transmitted via networks requires short term secure channels whereas data that is stored for the long term requires encryption techniques that are more resilient. Furthermore it has to be considered that encryption is not a onetime task to achieve confidentiality. As algorithms get outdated and broken, data might need to be re-encrypted with newer algorithms that provide secure confidentiality in a more recent technology. During all these processes encryption keys and other information needs to be preserved. The TIMBUS Security DSO supports this process by describing security features and attaching additional information such as encryption keys or passwords.

In general encryption is a conflicting concept when it comes to long term preservation as it is oppositional to the goal of preserving data. Encryption is used in order to hide data from unauthorized views. This also entails that it is much more difficult to treat encrypted data during the preservation process as it cannot be read easily. Therefore encryption is also a potential threat to any preservation activities (Storer, Greenan, & Miller, 2006). Preserving the passwords, cryptographic keys and the required software that can be used in order to decrypt the data is obviously very essential. Without this information the data could be lost and might not be restored again. The ontology we present allows describing the different components that are used for encrypting and decrypting the data.

The identity concept is closely related with the concepts of user identities and authentication and associated permissions. Identity management systems allow organizations to manage the accounts of users and systems as well as their identities in different contexts of the IT landscape, authentication schemes and authorization mechanisms. Such systems allow administering accounts in a centralized way, several implementations exist that could be used to extract this information and include it into the security ontology (Tracy, 2008).

Accountability is closely related to the concept of non-repudiation and denotes the principle that users can be made accountable for their actions, which cannot be denied because of the non-repudiation principle. This entails that it is essential keeping records of all relevant actions that have been performed by an actor within a system. The records must serve as evidence that provides traces to the responsible actor that was performing a specific action of interest (Stoneburner, 2001).

The compliance class serves as a reference point for different policies that might be relevant for the context of process execution.

The security objectives are not independent concepts, but are based upon each other. Confidentiality is only possible if integrity can be guaranteed. Vice versa integrity requires confidentiality, for instance in order to protect hash keys from being accessed by intruders. Also availability and accountability depend on each other and require confidentiality for protection and integrity for the completeness of the data (Stoneburner, 2001).

The classes that are available in the Security Ontology need to be related with each other in order to provide deeper knowledge. These relationships are denoted as object properties and they define how the entities depend on each other. Some of the properties are symmetric, meaning that they are available bidirectional. Figure 56 shows a simple example.

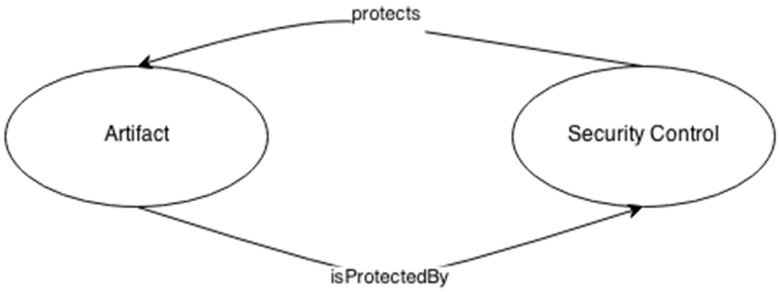


Figure 56: Class Relationships.

The following table shows the available predicates and provides a short description. Reading and interpreting the object properties follows the subject – object – predicate schema.

Table 8: Object Properties.

Object Property Name	Description
associatedWith	Relationship between the two classes exists.
composes composedOf	Classes form a compound of several entities.
defines definedBy	Class A defines Class B
describes describedBy	Class A describes Class B
encrypts isEncrypted	Security Control A encrypts artifact B
grants isGrantedBy	Class A grants permissions to class B (authorization mechanism C grants write permissions to agent D)
hasAccessTo isAccessedBy	Class A accesses class B (user C has access on machine D)
hasPermission	Class A has permissions with class B
implements isImplementedBy	Class A implements class B (e.g. LDAP implements authentication mechanism)
assigned isAssignedBy	Class A is assigns to class B ()
protects isProtectedy	Class A protects class B (security control C protects database D)
provides isProvidedBy	Class A provides class B (library C provides encryption mechanism D)
realizes	Class A realizes class B (authentication mechanism C

isRealizedBy	is realized by LDAP D)
regulates isRegulatedBy	Class A regulates class B (privacy policy C regulates confidentiality D)
requires isRequiredBy	Class A requires class B (password mechanism C requires password D)
revokes isRevokedBy	Class A revokes Class B (key revocation certificate C revokes key D)
uses isUsedBy	Class A uses Class B (audit C uses log file D)

The object properties presented in this table can be used to describe the relationships between that classes that are used in order to model security features of a process. They can also be used for creating the bridge between different ontologies, e.g. the DIO and the DSO.

4.6 Integrating the Security DSO into the TIMBUS DIO

The generic TIMBUS DIO can reference the TIMBUS Security DSO presented in this deliverable, thus the DSO can be seamlessly integrated into the DIO by using relationships between the two domains and tie them together. The integration of the Security DSO with the DIO is via ontology mapping, where some elements of the DSO are defined to be equivalent or subclasses of concepts in the DIO. There exist several reference points that enable the integration of the Security DSO into the TIMBUS DIO. The core concept which serves as central interface between the two ontologies is the Artifact class. The Security DSO artifact is a subclass of the DIO artifact class, as the Security DSO does not cover all of the concepts that are used in the DIO. However, this mapping allows creating the relationship between the Security DSO and the DIO and therefore allows describing the security properties of the relevant entities in detail. Further integration interfaces are the security domain class Human Resources and the Business Actor classes from the DIO respectively, which describe human factors. By utilizing the concepts provided from the security domain, the required competencies, skills and contracts of an actor can be described in an abstract way. Similarly, user roles can be mapped against business roles and tie the security requirements which are associated with a specific set of permissions. This mechanism allows associating constraints to user roles and equipping them with a fine grained set of privileges which enables them to fulfil their tasks. Table 9 shows the equivalence classes between the two ontologies that allow the integration of the DSO into the DSO.

Table 9: Mappings between the DIO and the Security DSO.

SECURITY DSO Classes	**TIMBUS DIO Classes**
Artifact	Artifact
Human Resources	Business Actor
User Role	Business Role

Furthermore several generic object properties such as isProtectedBy, isUsedBy or isEncryptedBy allow connecting existing concepts with their appropriate security features in a very flexible way. Note that not every element in the DSO has to be mapped, as some of them are used to describe information on those concepts that are mapped to the DIO in more detail. A detailed discussion on this integration can be found in (TIMBUS Consortium, 2013a). The mapping file for the equivalence classes of the Security DSO and the TIMBUS DIO can be found here: https://timbus.teco.edu/ontologies/DSOs/securityMapping.owl

4.7 Use Case Applications of the Security DSO

The main litmus test for the security DSO is whether it describes the security features of the use cases in a formal way such that all relevant information can be preserved. To begin with the description process, the following steps shown in Figure 57 need to be performed.

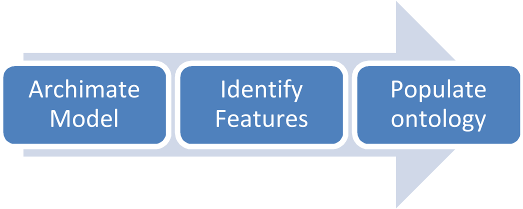


Figure 57: From Archimate to the Security Ontology.

The following Section 4.7.2 and Section 4.7.3 demonstrate how the Security Ontology has been applied to the use cases and how its fitness for purpose has been ensured.

In line with the evaluation objectives discussed in section the evaluation methodology is designed to answer three main questions:

1. Does the ontology fit the security properties of the use case?
2. Does the ontology cover all security properties of the use case?

Evaluation of these two questions will reveal, whether the ontology is suitable to describe the use case or whether it is unable to appropriately describe the security-related context of the use case. Secondly, it will also reveal whether the use case provides additional security-related context which is not yet covered by corresponding features of the ontology.

In order to complement the use case description in existence today, a third question is answered as part of the evaluation:

3. Does the use case description cover all security properties of the ontology?

Evaluation of this question not only provides an additional sanity check on the quality of match between ontology and use case it also provides additional information on where to supplement the use case documentation with regard to security-features of the use case.

4.7.1 Application to the use case

In order to answer the questioned outlined in section 4.7, the ontology needs to be applied to the use case. To achieve this, a simplified version of the Y-Model (Simon, 2010) was used, with the ontology providing the space of control attribute and the use case providing the space of control objects (cf. Figure 58).

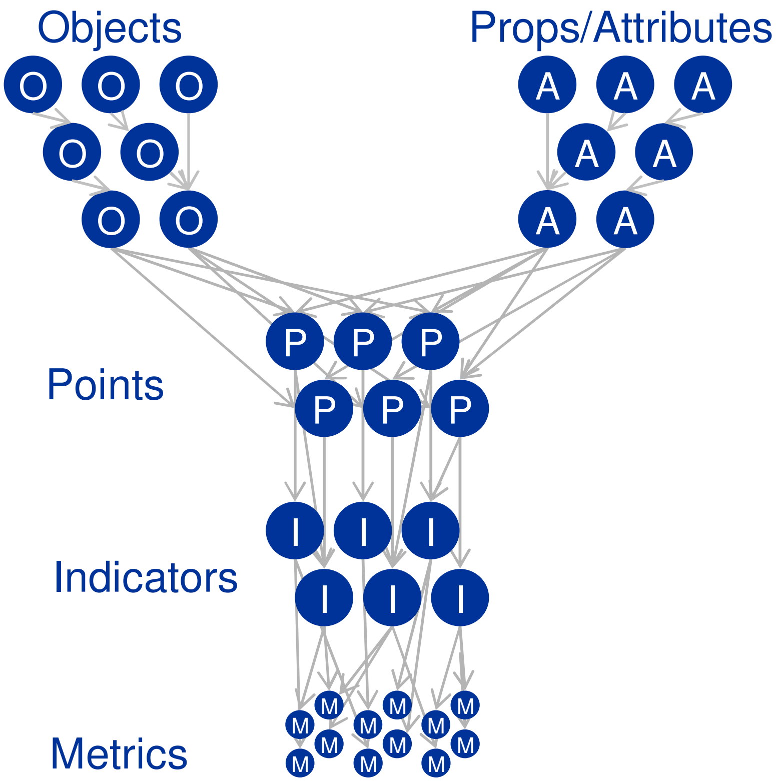


Figure 58: The Y-Model methodology used to ensure completeness and consistency of information capture.

Using this model, the spaces of objects and attributes are combined by creating the Cartesian product of both and reducing the resulting space by all irrelevant elements. Subsequently, for each remaining control point, indicators are sought which would point to adverse or undesired impacts on the combination of control attribute and control object. Here, impacts are clustered according to questions (1) – (3) as per section 4.7. In this simplified Y-Model, no metrics are used to quantify the adverse effects.

4.7.2 Application of the Security Ontology to the LNEC2-a Use Case

The Archimate model of the LNEC 2-a use case is shown in Figure 59. For simplicity reasons the image is limited to the infrastructure layer which has the most relevant features for the security ontology to extract.

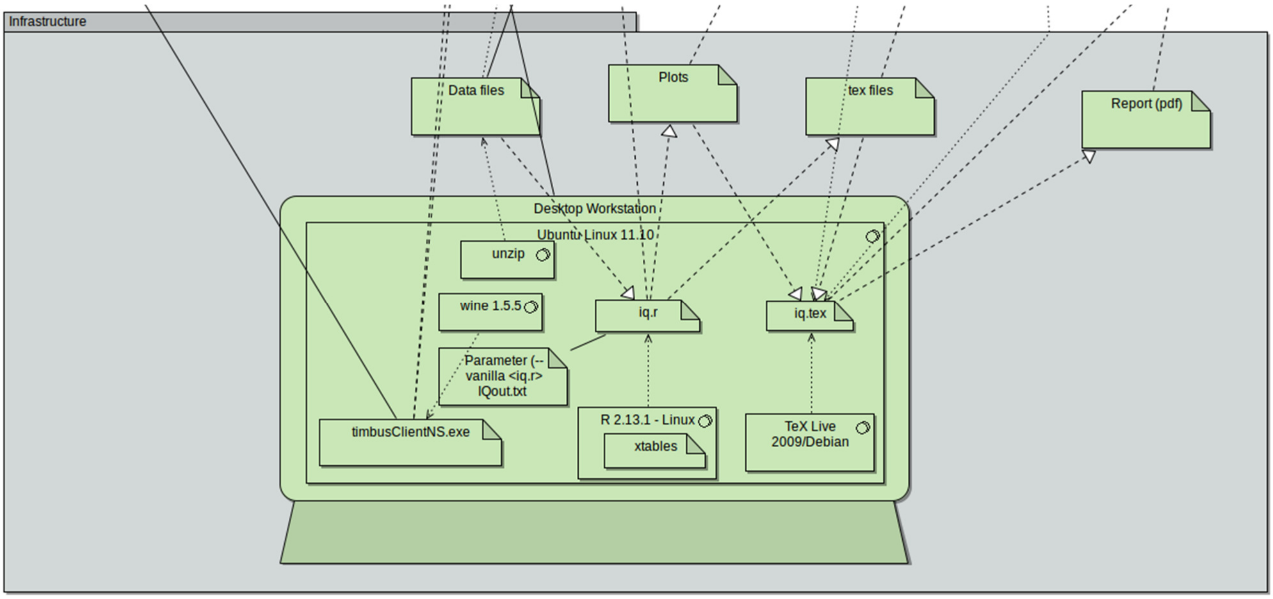


Figure 59: Archimate Infrastructure Model of LNEC 2-a for the Linux Workstation.

The user logs into the Ubuntu Linux workstation and executes the client which retrieves the data. Then, the execution of the R script and the Latex documents are generated and finally compiled into a PDF report. Various security features are used in the use case which can be described with the TIMBUS Security Ontology in detail. Picture Figure 60 shows the used classes and their relation.

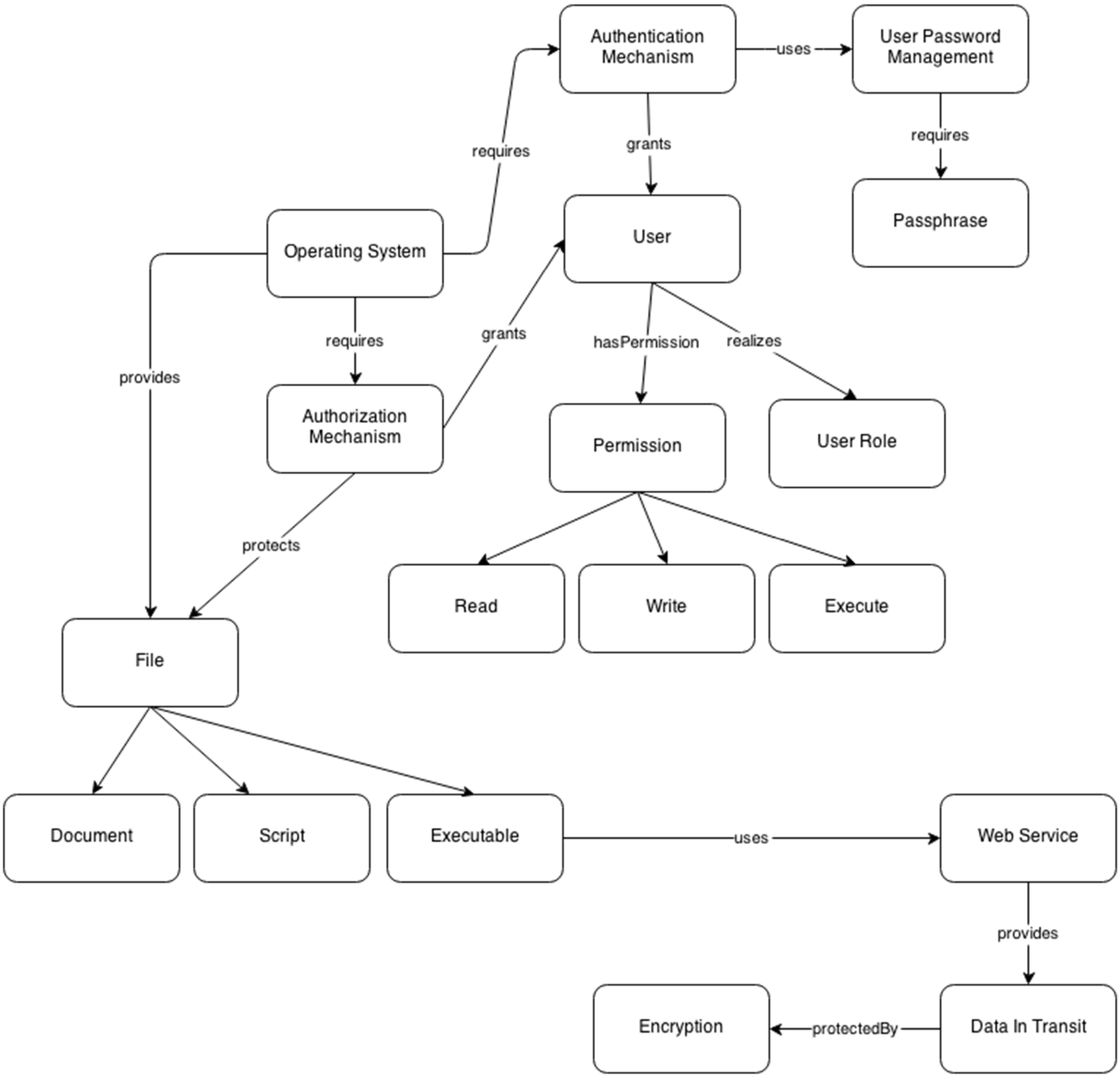


Figure 60: The Security Concepts Used in the LNEC 2-a Use Case.

Figure 60 shows an overview of the concepts used in the LNEC-2a use case in a high level view. In order to establish a mapping between the generic DIO and the Security Domain, we exported the Archimate model of the use case into the OWL file and imported the Security DSO into the use case ontology. By instantiating individuals of the appropriate classes and using the object property assertions, we could model the process. The following Figure 61 shows the use case described with the TIMBUS Securuity Ontology. The model is simplified in order to demonstrate the application of the ontology on the use case.

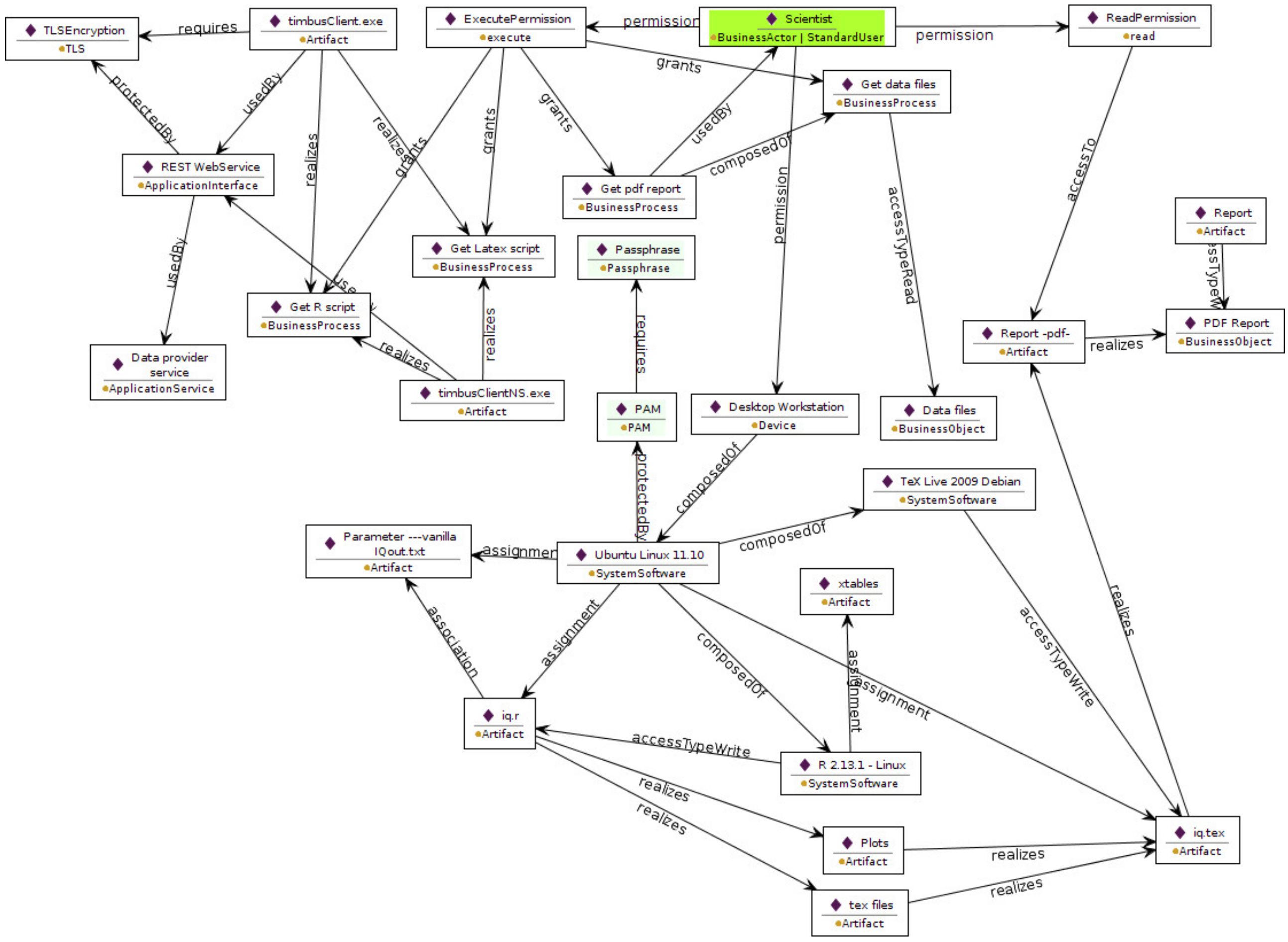


Figure 61 The LNEC 2 Use Case with Individuals

The security DSO allows annotating individuals from the DIO or e.g. the Software DSO with security information in a very fine grained level. Thus it allows to model permission rights for users and services, login information as well as encryption standards for data in transit. The individuals present from the Archimate model can be related to the security concepts from the DSO by using the provided object properties.

4.7.3 Application of the Security Ontology to the WP8 “eHealth” use case

The following figure illustrates the control objects used for the evaluation of the Drugs use case.

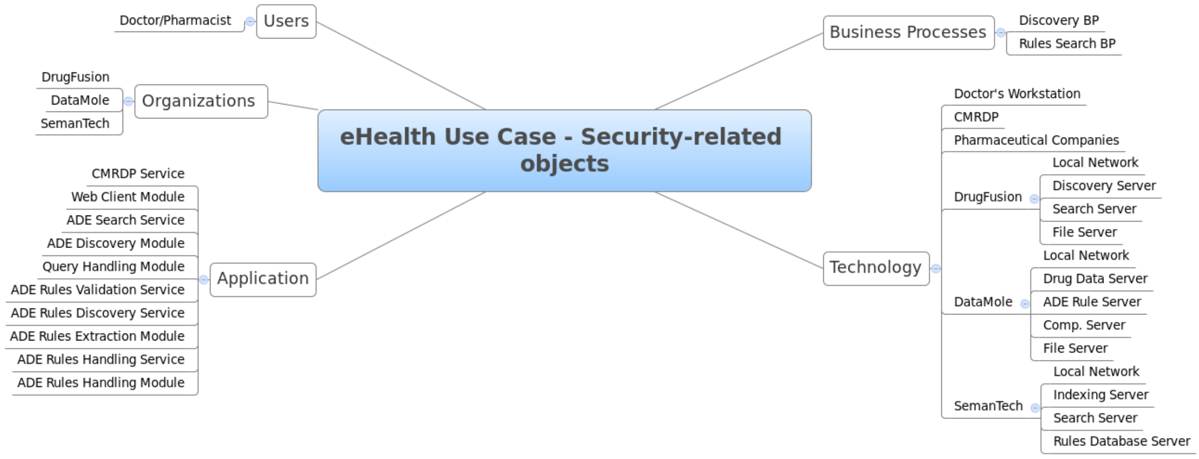


Figure 62: Control objects derived from the Drugs use case

In order to apply the security DSO to the eHealth use case, we exported the Archi model into its OWL representation and annotated the services from the infrastructure layer with details about the security features that have been implemented.

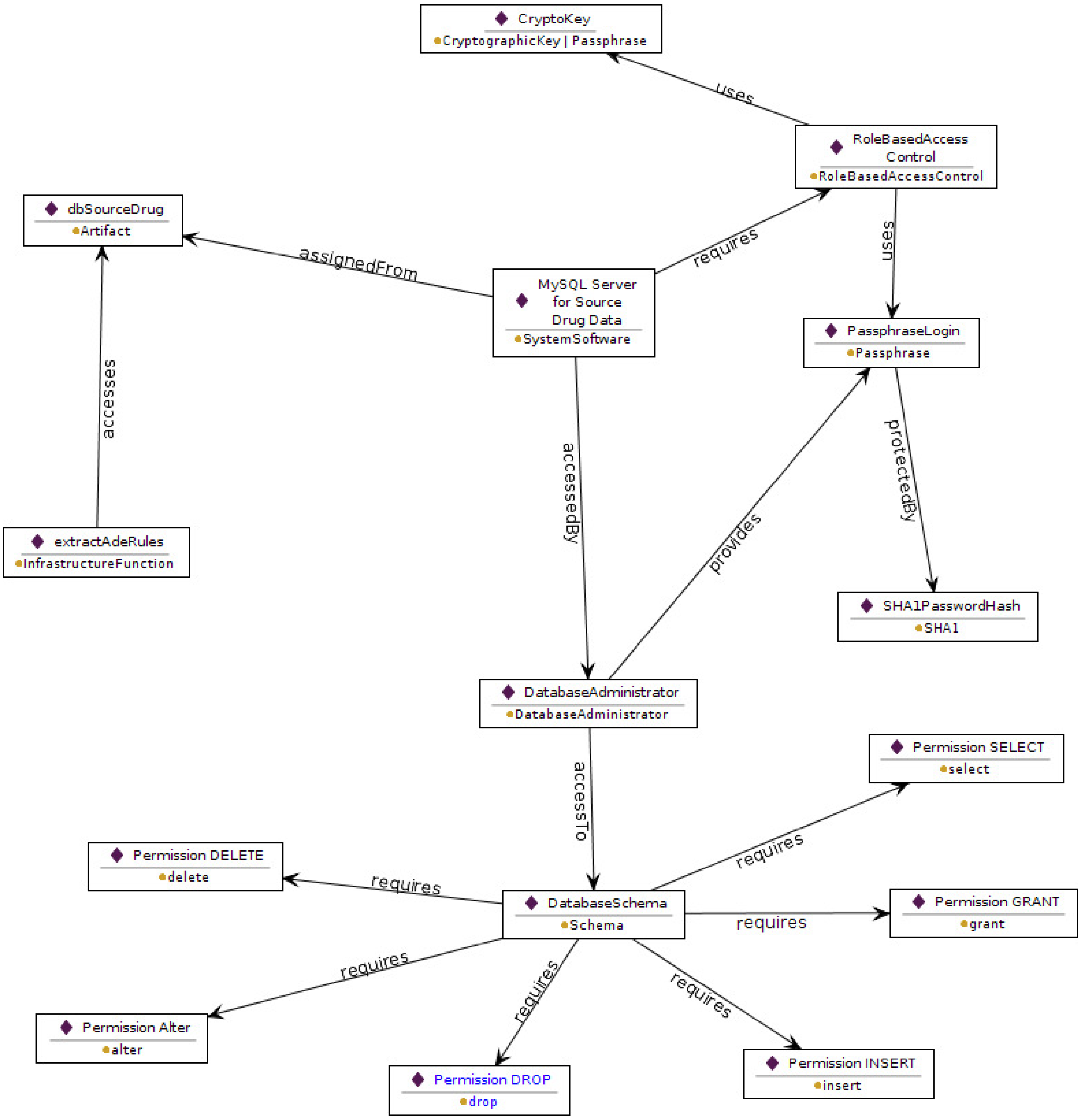


Figure 63 The Application of the Security DSO to the eHealth Use Case

The application of the Security DSO is depicted Figure 63, which shows a small subset of the eHealth use case. This visualization of the individuals shows how the different DSOs can be mapped my importing all required ontologies into the DIO. The examples show the MySQL Server instance handling the data about drugs. It was augmented by security details about its administrator user, who either needs a cryptographic key or a password in order to login into the system. The database implements a role based access control system which grants the authenticated user with several rights for a database schema.

The ontology matches the security-related aspects of the use case well. All security aspects of the use case can be modelled using the security ontology and both ontology classes as well as their relations fit to the security features of the use case.

4.7.4 Summary the Security Ontology Applications

The description of the security features that are provided by the ontology fulfil several purposes and allow different types of applications. Firstly, the ontology allows generating a visualization of the concepts which are used in a use case and depicts their relationships in a clear manner. Secondly the ontology allows attaching concrete metadata and additional information about the security processes directly to the model and therefore increases the preservability of the process. By attaching properties such as cryptographic keys or passwords directly to the model, their long term usage can be simplified. Thirdly the ontology provides a formal model that can be queried for retrieving answers regarding the security implementations used by a process. For instance the ontology allows retrieving all users that have had access to a specific file or service within a complex process. Other applications are the detection of obsolete security implementations that cannot provide the security level that a process requires. Hence the ontology provides knowledge which subservices use a specific implementation and therefore requires maintenance.

5 Conclusions and outlook

In this deliverable we presented the outcomes of research on tasks:

- T4.6 Process and Method for Validation of Preserved Business Processes,
- T4.7 Security and Authorisation Process for Preservation and Redeployment,
- T4.8 Process and Method for Redeployed Business Processes Verification.

The outcomes of tasks T4.6 and T4.8 were discussed in Section 3. The outcomes of task T4.7 were discussed in Section 4. We provide the conclusions and outlook using the same grouping, i.e. firstly discussing the verification and validation and then the security. The outcomes of this deliverable will be used by WP7 and WP8 when applying the TIMBUS preservation framework.

5.1 Verification and Validation

In this deliverable we have presented a set of concepts enabling verification and validation of preserved and redeployed business processes:

- VFramework – framework for verification of preserved process,
- VPlan – ontology for storing verification data,
- VHelper – proof of concept tool automating verification and validation process,
- SPARQL queries – set of queries allowing to validate and present collected information.

The proposed solution was evaluated on two use cases. The first use case stems from the WP7 (TIMBUS Consortium, 2013c)and deals with an open source workflow. The application of the VFramework steps, creation of the VPlan, usage of the VHelper, and validation of data using SPARQL queries were presented for the preservation phase of the use case. The same steps were also demonstrated for the use case from the WP8 (TIMBUS Consortium, 2012) which deals with a data transformation process in the domain of civil engineering. Furthermore, for this use case we have simulated the redeployment phase by migration to another substantially different environment. For the purpose of redeployment, the process had to be re-engineered and adjusted to work in the new environment. In order to perform the comparison of metrics we had to implement tools which automate the metric extraction and comparison – the VFramework comparators were implemented. The proposed solution was applicable in both of the considered use cases and we were able to reliably verify the processes and validate the collected information and data.

Future work should focus on further automation of the verification process. The tools needed for extraction and comparison of measurements taken for significant properties need to be created. Furthermore, application to different cases and different redeployment scenarios is needed to evaluate the solution scalability. Possible integrations of the VPlan with existing solutions in the area of digital preservation and a broad scale application of the VFramework can improve substantially the preservability of not only business but also scientific processes.

5.2 Security

Security is an essential non-functional property that requires close attention during all three phases of the TIMBUS Project. Several different aspects constitute a secure system and it is not sufficient preserving the status quo. The security measures introduced into systems need to satisfy security requirements such as confidentiality, integrity or the authenticity of sensitive data. It is clear that this data requires protection during the preservation phase of a business process and also during its redeployment. Preserving the security of a business process is a challenging task, as many of these measures counteract preservation activities. Security measures are implemented with the intention to limit access by design and should be difficult to circumvent in the first place. Preservation in contrast aims to keep information accessible in the long term. Preserving security features is located in the centre of this field of tension, as it has to satisfy both aspects. Therefore security can be considered as an especially interesting aspect in the context of digital preservation, because they add an additional layer of complexity.

The goal of the work described in this deliverable was to tackle the challenges of preserving systems in a secure way by describing the security features that are required for the safe execution of a business process in a precise way. For achieving this goal we designed a domain specific ontology that is capable of mapping the security knowledge of a business process and therefore describe components and their associated security measures in detail. We identified more than 150 aspects of security and modelled them into entities, which we then out into relation. Our selection of relationships between these entities allows flexible yet precise descriptions of the security features that are present in a business process. The ontology we provided allows identifying; describing, querying and preserving this knowledge for the long term and it can be seamlessly integrated into the generic TIMBUS DIO and therefore contributes a holistic view on business processes and their properties. The benefit of this work is that it allows to domain experts to analyse the security features which are used in business processes, extract their properties and features, and model these aspects. The conserved knowledge provided by the ontology will allow reacting on future developments in the sector of secure computing and therefore enable secure redeployment of sensitive business processes.

A SPARQL queries for VPlan

```
PREFIX dio: <http://timbus.teco.edu/ontologies/DIO.owl#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX vplan: <http://timbus.teco.edu/ontologies/VPlan#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

################# Validation if all needed properties are added to the individual  #################

#change Metric to the Class you want to check and regex "SP1M1" in filter to match individual
        SELECT distinct ?properties
        WHERE
        {
                vplan:Metric rdfs:subClassOf+/owl:onProperty ?properties. #change for each class

                OPTIONAL {
                        ?x a vplan:Metric.      #change for each class
                        ?x ?prop ?y.
                        ?prop a ?z
                        FILTER(regex(str(?x), "SP1M1")) # for each individual perform the check
                        FILTER(?prop = ?properties)
                }
                FILTER (!bound(?prop)) .
        }

################# Significant property listing #################

#List significant properties
select *  where {
?proc a vplan:SignificantProperty.
}

#Show description for each signifacant property
select *  where {
?proc a vplan:SignificantProperty.
OPTIONAL{?proc vplan:hasTextDescription ?description.}
}

#Show list of metrics for each significant property
select *  where {
?proc a vplan:SignificantProperty.
OPTIONAL{?proc vplan:isMeasuredBy ?metric.}
}

#Show list of scenarios for each signifacant property
select *  where {
```

```
?proc a vplan:SignificantProperty.
OPTIONAL{?proc vplan:appliesToScenario ?scenario.}
}

################# Metric listings #################

#Show description for each metric
select *  where {
?proc a vplan:Metric.
OPTIONAL{?proc vplan:hasTextDescription ?description.}
}

#Show Metric Target Value for each metric
select * where {
?metric a vplan:Metric.
OPTIONAL{?metric vplan:hasMetricTargetValue ?targetValue.}
}

#Show Metric Operator for each metric
select * where {
?metric a vplan:Metric.
OPTIONAL{?metric vplan:hasMetricTargetOperator ?operator.}
}

#Show Capture Processes for each metric
select * where {
?metric a vplan:Metric.
OPTIONAL{?metric vplan:hasCaptureProcess ?capture.}
}

#Show Capture Processes Instance for each metric
select * where {
?metric a vplan:Metric.
OPTIONAL{?metric vplan:hasCaptureProcessInstance ?instance.}
}

#Show artifacts used for metric computation of each metric
select ?metric (str(?ComputationArtifact) as ?usedForComputation) where {
?metric a vplan:Metric.
OPTIONAL{?metric vplan:isUsedForMetricComputation ?computation.
?computation rdfs:label ?ComputationArtifact
}
}

################# Capture Process listings #################

#Show list of Capture Processes and elements they are composed of
select distinct ?proc  (STR(?elementsLab) as ?composedOf) where {
```

```
?proc a vplan:CaptureProcess.
OPTIONAL{?proc dio:composedOf ?elements.
?elements rdfs:label ?elementsLab.}
}

#Show list of Capture Processes and Artifacts they have
select distinct ?proc (STR(?label) as ?artifactLabel) where {
?proc a vplan:CaptureProcess.
OPTIONAL{?proc vplan:hasArtifact ?artifact.
?artifact rdfs:label ?label.}
}

#Show list of Capture Processes and Instances they have
select *  where {
?proc a vplan:CaptureProcess.
OPTIONAL{?proc vplan:hasInstance ?instance.}
}

################# Capture Process Instance listings #################

#Show Description for each Capture Process Instance
select *  where {
?proc a vplan:CaptureProcessInstance.
OPTIONAL{?proc vplan:hasTextDescription ?description.}
}

#Show Capture Process Data for each Capture Process Instance
select *  where {
?proc a vplan:CaptureProcessInstance.
OPTIONAL{?proc vplan:hasInstanceData ?data.}
}

################# Capture Process Data listings #################

#Show location for each Capture Process Data
select *  where {
?proc a vplan:CaptureProcessData.
OPTIONAL{?proc vplan:isLocatedAt ?location.}
}

#Show for which element is the data collected (realizes connection) [Jena would show inferred results]
select ?proc (str(?elementLab) as ?artifact)  where {
?proc a vplan:CaptureProcessData.
OPTIONAL{?proc dio:realizes ?element.
?element rdfs:label ?elementLab}
}
```

```
################# Redeployment Scenario listings #################

#Show Description for Redeployment Scenario
select * where {
?proc a vplan:RedeploymentScenario.
OPTIONAL{?proc vplan:hasTextDescription ?description.}
}

#Show Instances for Redeployment Scenario
select * where {
?proc a vplan:RedeploymentScenario.
OPTIONAL{?proc vplan:hasInstance ?instance.}
}

#Show Artifacts for Redeployment Scenario
select * where {
?proc a vplan:RedeploymentScenario.
OPTIONAL{?proc vplan:hasArtifact ?artifact.}
}

#Show elements of which the Redeployment Scenario is composed
select * where {
?proc a vplan:RedeploymentScenario.
OPTIONAL{?proc dio:composedOf ?composed.}
}

################# Redeployment Scenario Instance listings #################

#Show Description for each Redeployment Scenario Instance
select *  where {
?proc a vplan:RedeploymentScenarioInstance.
OPTIONAL{?proc vplan:hasTextDescription ?description.}
}

#Show Redeployment Scenario Data for each Redeployment Scenario Instance
select *  where {
?proc a vplan:RedeploymentScenarioInstance.
OPTIONAL{?proc vplan:hasInstanceData ?data.}
}

################# Redeployment Scenario Data listings #################

#Show location for each Redeployment Scenario Data
select *  where {
?proc a vplan:RedeploymentScenarioData.
OPTIONAL{?proc vplan:isLocatedAt ?location.}
}
```

```
#Show for which element is the data collected (realizes connection)
select ?proc (str(?elementLab) as ?artifact)  where {
?proc a vplan:RedeploymentScenarioData.
OPTIONAL{?proc dio:realizes ?element.
?element rdfs:label ?elementLab}
}

################# Auxiliary Resources listings #################

#List resources and thier descriptions
select *  where {
?proc a vplan:AuxiliaryResource.
OPTIONAL{?proc vplan:hasTextDescription ?description.}
}

#List resources and their location
select *  where {
?proc a vplan:AuxiliaryResource.
OPTIONAL{?proc vplan:isLocatedAt ?location.}
}
```

B LNEC Security questions

- What systems with different authorization mechanisms are available

We mainly have to authorization mechanisms. (a) database authentication; (b) app authentication. We share the app authentication between several applications and components. E.g. the authentication mechanism is the same for remote applications (e.g., portable data terminals) and the main information system (gestBarragens). We have a 1-to-1 mapping between an application user and a database user.

- What interfaces exist between these systems?

The main information system (gestBarragens) stores app users/passwords and also the corresponding database-user/password for each app user.

- Do Web services require authorization?

Yes

- What permission or authorization models do you currently use (e.g. Role Based Access Control, ACL, LDAP,...)?

Role Based

- What authorization systems or models are used locally and remote (Web services, databases)? What are the interfaces where authorization is verified?

Both app and database authorization are verified by functional operation (e.g., if we want to upload a set of data from a file, the authorization is just verified one time, and not at row-level).

- Are there any roles defined, that are associated with specific access permissions? If so, are these roles identical with the stakeholders identified in D8.1 (IT Manager, IT Technician, FU Researcher, ...) and are there any further refinements available?

The roles are similar. In fact, in the scope of our application we have one role for IT Manager. In our case, the IT Technician is just responsible by operating system configuration, network, firewall, etc. Then, we have three main levels:

- Manager (maps to FU Researcher)
- Technician (maps to FU technician)
- Registered user (any user that is registered in the system, but does not possess any specific role)

There is also a distinction between users that are internal to LNEC or external users (e.g., structure owners) The critical detail is that a Manager is a "Structure Manager" and a Technician is a structure Technician. in other words, these roles are not generic to the overall system, but a role that a user possess for a specific structure. For instance, userX can be a manager for DamA, a technician for DamB and does not have a role for DamC (equivalent to registered user).

- Are there formal descriptions of these roles available?

We have descriptions in the technical documentation, but it is written in Portuguese

- Is there an organization wide policy?

n.a.

- How many members of each role are there?

That's a difficult question, due to the partition by structure. Overall numbers, the LNEC instance has 101 users, while the EDP instance has 71 users.
At LNEC, 44 users are managers of at least one structure.
At EDP, 7 users are managers of at least one structure.

At LNEC, 30 users are managers of at least one structure.

At EDP, 63 users are technicians of at least one structure.

- Does there exist a security policy?

n.a.

- Are the business processes explained in D8.1 accompanied by authorization mechanisms and differentiate between individual users or roles?

authorization is done by role. We have some use cases where the
authorization distinguishes between internal and external users, but
that is not the case for the business processes addressed in D8.1.

- What method is used to authenticate end users to the system? What are the components users have to authenticate against? How are users' authorizations determined and enforced?

Encrypted username and password, verified at server.

- Do you use encryption or signatures at any point? What type of encryption is used? How is it configured and deployed?

We have some components where we use encryption, but it is not fully
disseminated yet. Anyway, when we use encryption, the algorithms are:
- RSA for public keys
- Rijnael for simetric algorithms
- SHA1 for hashes.

That's the current state, but we plan to improve it.

- Do you log interactions with the systems (user log, authentication log, error log)

We have two types of logs:
- App logs, where we log critical operations (we log in database tables and/or files)
- Database logs. These are the critical ones. Since we have a one-to-one mapping between app users and database users, we can link all actions to individual users.

- Is sensitive data contained in logs?

Yes.

- Can logs link actions to individual users?

Yes.

- How is access to the logs controlled?

Only admins have access database and file logs. Critical user
operations are logged in database and we have a component (similar to
an email client) where users can check the logs of their operations.

- How long are logs retained?

We try to preserve them forever

- Do you record any metadata for describing users, roles and hierarchies?

Technical documentation

- Is there any change management implemented (change propagation of authorization/user permission data)?

No, but we could get it from database logs.

- Do you have Digital Rights Management in place?

no

- Is there any data or systems involved in the use case, which has restricted access to specialized users?

yes

- Is there data which has to be safeguarded after redeployment and must not be read by unauthoriized personnel?

yes

- Are there different database views for the various roles or can everyone see/modify all records?

yes, it is different by role per structure.

C VFrame Comparators

C.1 Objective

This document will describe the implementation, usage and sample outputs of the comparator tools (png, pdf and latex) developed as part of Verification and Validation Framework (VVF).

C.2 Image (PNG) Comparator

The objectives of this comparator is

- To input two different image files of same frame size
- Extract key features from those two images
- Construct third image file with differences only pixels
- Produce summary table of differences comparing extracted key features

C.2.1 Implementation

Image comparator tool was developed in Java utilising the power of ImageMagick[26] through img4java[27] API. ImageMagick is a software suite to create, edit, compose, or convert bitmap images. It can read and write images in a variety of formats (over 100) including DPX, EXR, GIF, JPEG, JPEG-2000, PDF, PNG, Postscript, SVG, and TIFF. Use ImageMagick to resize, flip, mirror, rotate, distort, shear and transform images, adjust image colours, apply various special effects, or draw text, lines, polygons, ellipses and Bézier curves. It's a cross platform compatible and is free software delivered as a ready-to-run binary distribution or as source code that you may use, copy, modify, and distribute in both open and proprietary applications. It is distributed under the Apache 2.0 license.

On other hand Img4java is a pure java interface to ImageMagic command line, and are stable quite stable, so the comparator java tool should work across many versions of IM and various OS types without need of JNI[28]. Img4java also provides a better OO interface (the "language" of the IM-command line with it's postfix-operation notation translates very easily into OO-notation).

[26] ImageMagick suite: http://www.imagemagick.org/script/index.php

[27] Java API for Image Magic: http://im4java.sourceforge.net/

[28] Java Native Interface: http://en.wikipedia.org/wiki/Java_Native_Interface

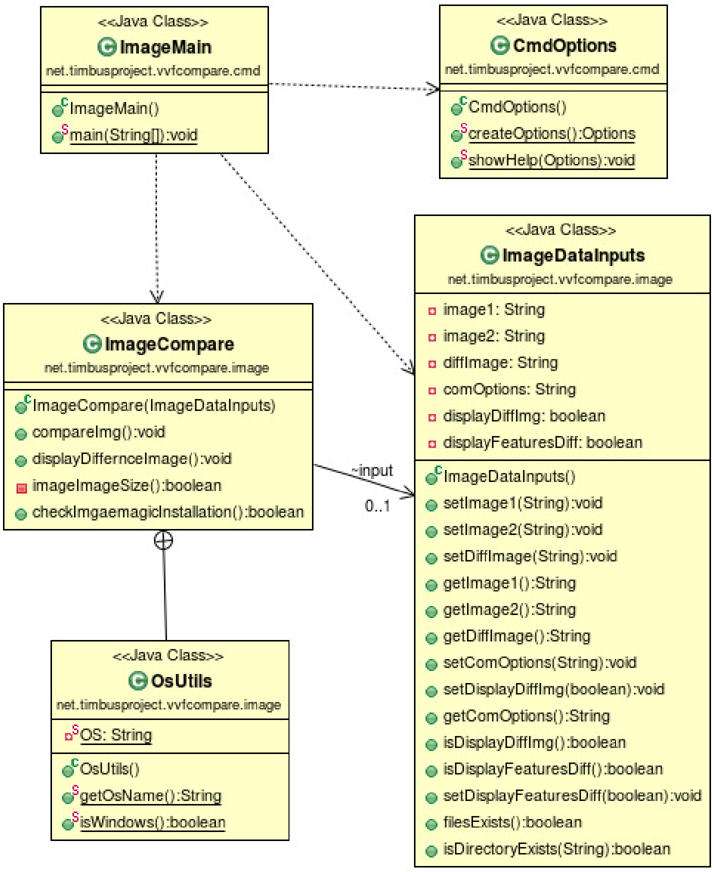
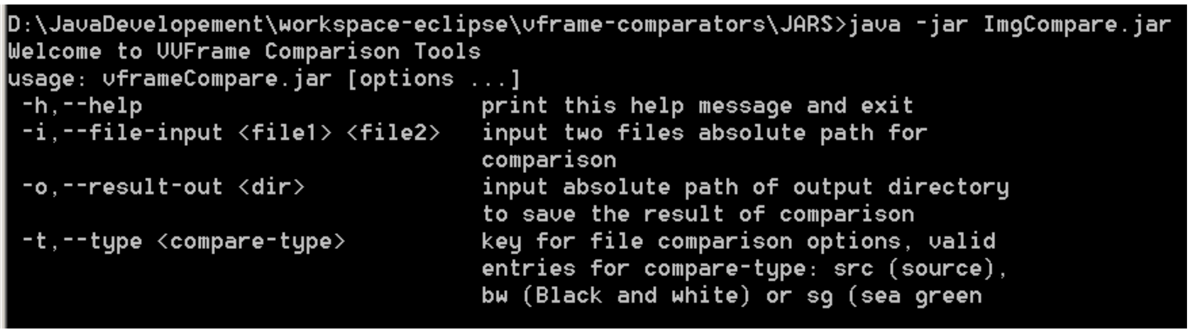


Figure 5-1: Image Comparator Implementation Java Class Diagram.

C.2.2 Examples

Usage:



```
D:\JavaDevelopement\workspace-eclipse\vframe-comparators\JARS>java -jar ImgCompare.jar
Welcome to VVFrame Comparison Tools
usage: vframeCompare.jar [options ...]
 -h,--help                       print this help message and exit
 -i,--file-input <file1> <file2>  input two files absolute path for
                                 comparison
 -o,--result-out <dir>           input absolute path of output directory
                                 to save the result of comparison
 -t,--type <compare-type>        key for file comparison options, valid
                                 entries for compare-type: src (source),
                                 bw (Black and white) or sg (sea green
```

Figure 5-2 : Image Compare Usage Screen capture.

Script:

```
java -jar ImgCompare.jar -i test-images\test_image_1.jpg test-images\test_image_2.jpg -
o img-outputs -t src
```

Outputs:

Table 10: Sample Image Comparison Output Table.

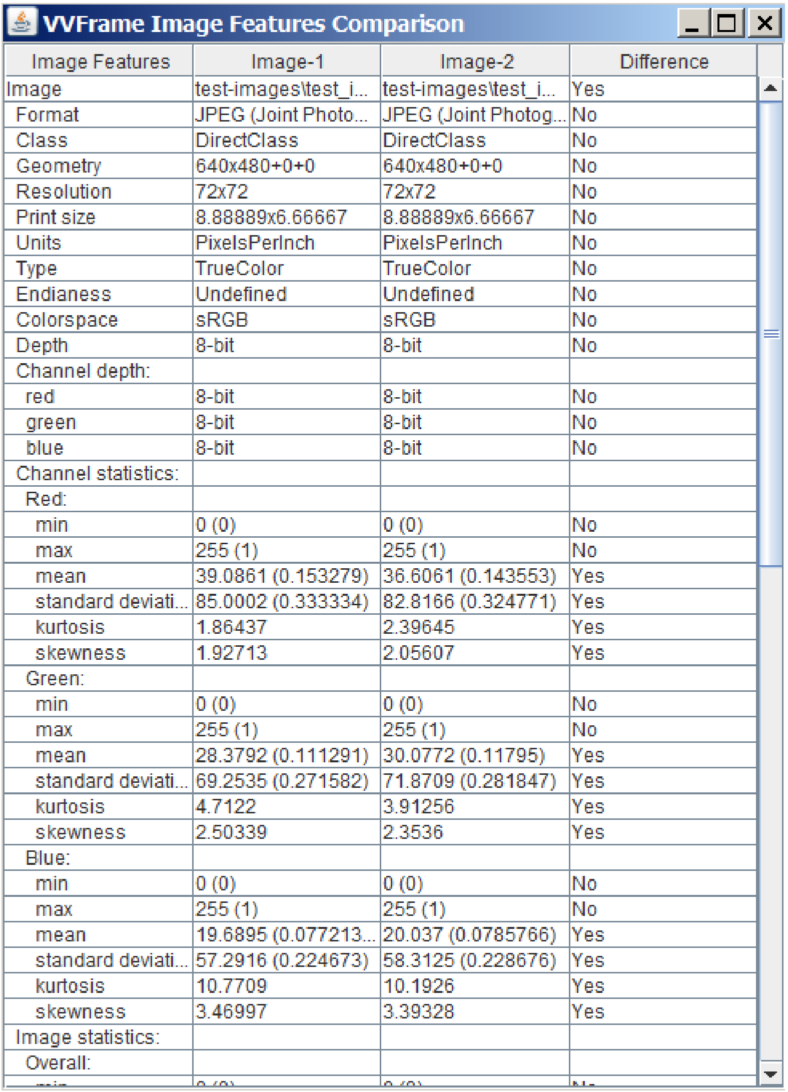


Image Features	Image-1	Image-2	Difference
Image	test-images\test_i...	test-images\test_i...	Yes
Format	JPEG (Joint Photo...	JPEG (Joint Photog...	No
Class	DirectClass	DirectClass	No
Geometry	640x480+0+0	640x480+0+0	No
Resolution	72x72	72x72	No
Print size	8.88889x6.66667	8.88889x6.66667	No
Units	PixelsPerInch	PixelsPerInch	No
Type	TrueColor	TrueColor	No
Endianess	Undefined	Undefined	No
Colorspace	sRGB	sRGB	No
Depth	8-bit	8-bit	No
Channel depth:			
red	8-bit	8-bit	No
green	8-bit	8-bit	No
blue	8-bit	8-bit	No
Channel statistics:			
Red:			
min	0 (0)	0 (0)	No
max	255 (1)	255 (1)	No
mean	39.0861 (0.153279)	36.6061 (0.143553)	Yes
standard deviati...	85.0002 (0.333334)	82.8166 (0.324771)	Yes
kurtosis	1.86437	2.39645	Yes
skewness	1.92713	2.05607	Yes
Green:			
min	0 (0)	0 (0)	No
max	255 (1)	255 (1)	No
mean	28.3792 (0.111291)	30.0772 (0.11795)	Yes
standard deviati...	69.2535 (0.271582)	71.8709 (0.281847)	Yes
kurtosis	4.7122	3.91256	Yes
skewness	2.50339	2.3536	Yes
Blue:			
min	0 (0)	0 (0)	No
max	255 (1)	255 (1)	No
mean	19.6895 (0.077213...	20.037 (0.0785766)	Yes
standard deviati...	57.2916 (0.224673)	58.3125 (0.228676)	Yes
kurtosis	10.7709	10.1926	Yes
skewness	3.46997	3.39328	Yes
Image statistics:			
Overall:			

C.3 PDF Document Comparator

The objectives of this comparator is

- To input two different pdf documents
- Construct third pdf file with differences highlighted using diff-pdf[29]
- Extract pdf documents metadata and fixities and produce comparison table
- Extract texts from pdf files and compare texts and then output them in html file format by highlighting differences in texts.

[29] Diff-pdf tools for visual pdf comparison: https://github.com/vslavik/diff-pdf

C.3.1 Implementation

PDF Comparator is implemented using following third party projects and libraries.

1. Diff-pdf: To visually compare two pdf documents and produce third pdf document highlighting the differenced. Diff-pdf source code available in github under GPL license and it is written in c/c++ so code should be compiled to target Operating system (windows, linux and etc.) into to run as standalone application.

   Source: https://github.com/vslavik/diff-pdf

   Precompiled binaries for windows:
   http://www.tt-solutions.com/downloads/diff-pdf-2012-02-28.zip

   Pdf comparator provides wrapper to diff-pdf tool to use within VFramework project.

2. Apache Tika[30]: This toolkit detects and extracts metadata and structured text content from various documents using existing parser libraries. This toolkit is used to generate various metadata information from pdf both files and comparison table is populated.

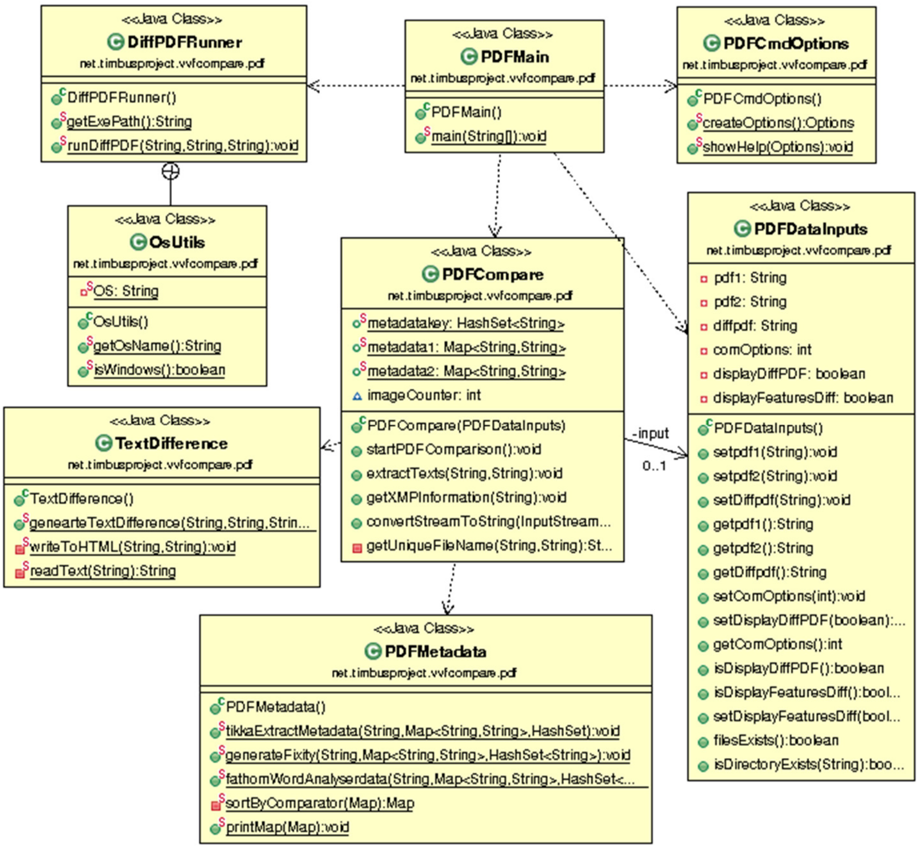
3. Apache PdfBox[31]: This is an open source Java tool for working with PDF documents. This project allows creation of new PDF documents, manipulation of existing documents and the ability to extract content from documents. This library is used to extract text contents from pdf file for comparison.

4. google-diff-match-patch[32] : The Diff Match and Patch libraries offer robust algorithms to perform the operations required for synchronizing plain text. The diff, match and patch algorithms in this library are **plain text only**. These libraries are being used to compare the extracted text and produce difference in html document.

[30] Apache Tika: https://tika.apache.org/

[31] Apache PDF Box: http://pdfbox.apache.org/

[32] Google diff match patch: https://code.google.com/p/google-diff-match-patch/

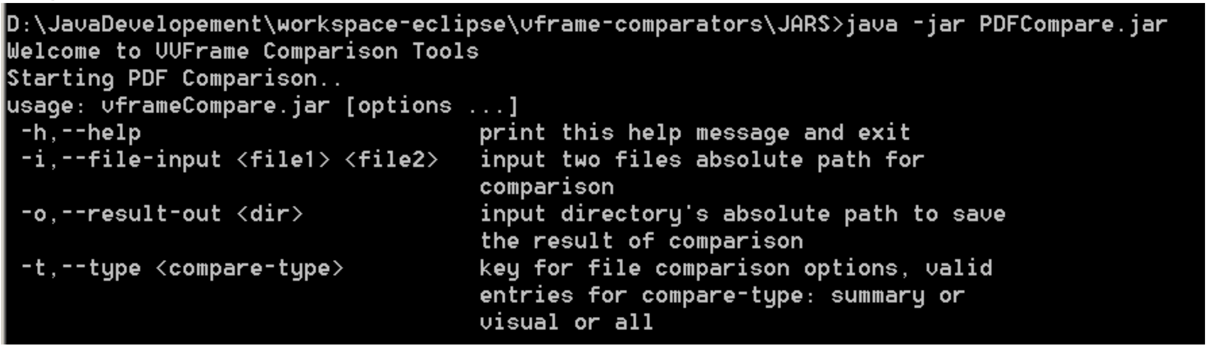


<<Java Class>>
DiffPDFRunner
net.timbusproject.vvfcompare.pdf
DiffPDFRunner()
getExePath():String
runDiffPDF(String,String,String):void
<<Java Class>>
PDFMain
net.timbusproject.vvfcompare.pdf
PDFMain()
main(String[]):void
<<Java Class>>
PDFCmdOptions
net.timbusproject.vvfcompare.pdf
PDFCmdOptions()
createOptions():Options
showHelp(Options):void
<<Java Class>>
OsUtils
net.timbusproject.vvfcompare.pdf
OS: String
OsUtils()
getOsName():String
isWindows():boolean
<<Java Class>>
PDFCompare
net.timbusproject.vvfcompare.pdf
metadatakey: HashSet<String>
metadata1: Map<String,String>
metadata2: Map<String,String>
imageCounter: int
PDFCompare(PDFDataInputs)
startPDFComparison():void
extractTexts(String,String):void
getXMPInformation(String):void
convertStreamToString(InputStream...
getUniqueFileName(String,String):St...
<<Java Class>>
PDFDataInputs
net.timbusproject.vvfcompare.pdf
pdf1: String
pdf2: String
diffpdf: String
comOptions: int
displayDiffPDF: boolean
displayFeaturesDiff: boolean
PDFDataInputs()
setpdf1(String):void
setpdf2(String):void
setDiffpdf(String):void
getpdf1():String
getpdf2():String
getDiffpdf():String
setComOptions(int):void
setDisplayDiffPDF(boolean):...
getComOptions():int
isDisplayDiffPDF():boolean
isDisplayFeaturesDiff():bool...
setDisplayFeaturesDiff(bool...
filesExists():boolean
isDirectoryExists(String):boo...
-input
0..1
<<Java Class>>
TextDifference
net.timbusproject.vvfcompare.pdf
TextDifference()
genearteTextDifference(String,String,Strin...
writeToHTML(String,String):void
readText(String):String
<<Java Class>>
PDFMetadata
net.timbusproject.vvfcompare.pdf
PDFMetadata()
tikkaExtractMetadata(String,Map<String,String>,HashSet):void
generateFixity(String,Map<String,String>,HashSet<String>):void
fathomWordAnalyserdata(String,Map<String,String>,HashSet<...
sortByComparator(Map):Map
printMap(Map):void

Figure 5-3: PDF Comparator Tool Class diagram.

C.3.2 Examples

Usage:



```
D:\JavaDevelopement\workspace-eclipse\vframe-comparators\JARS>java -jar PDFCompare.jar
Welcome to VVFrame Comparison Tools
Starting PDF Comparison..
usage: vframeCompare.jar [options ...]
 -h,--help                        print this help message and exit
 -i,--file-input <file1> <file2>  input two files absolute path for
                                  comparison
 -o,--result-out <dir>            input directory's absolute path to save
                                  the result of comparison
 -t,--type <compare-type>         key for file comparison options, valid
                                  entries for compare-type: summary or
                                  visual or all
```

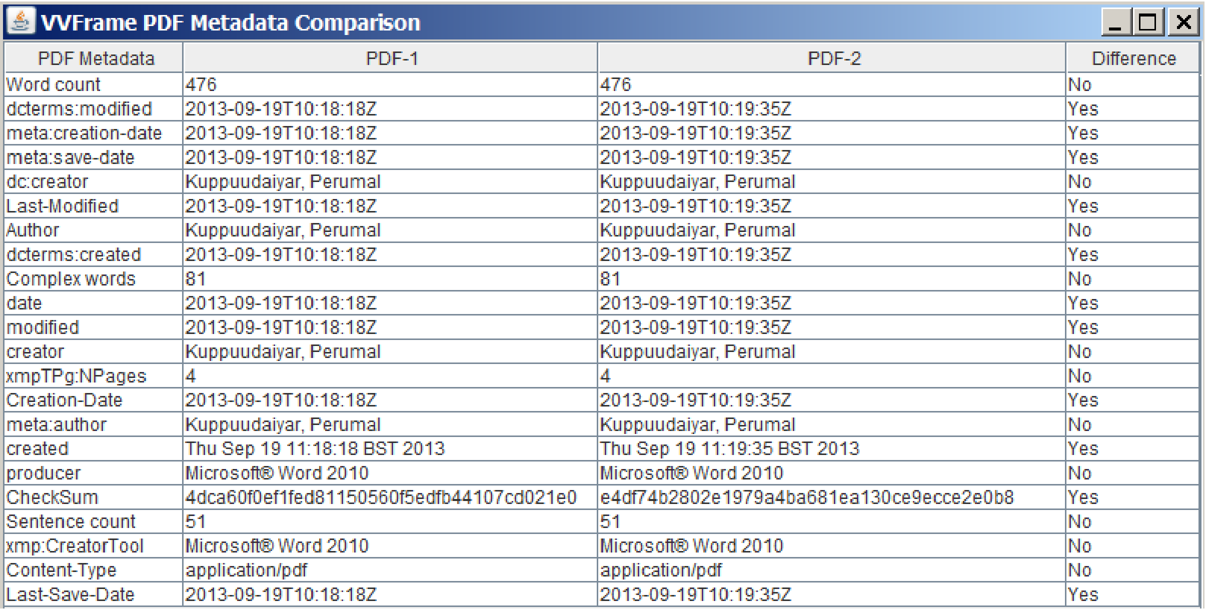
Figure 5-4: Screen capture of PDF Comparator Command line usage.

Script:

java -jar PDFCompare.jar -i test-documents\testword1.pdf test-documents\testword2.pdf -o pdf-outputs -t all

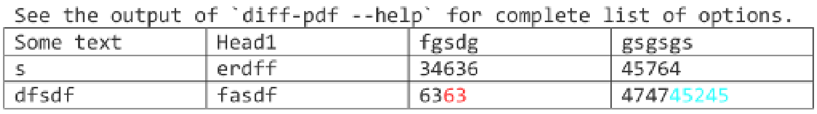
Output:

Table 11: PDF Comparison Sample Output.



VVFrame PDF Metadata Comparison

PDF Metadata	PDF-1	PDF-2	Difference
Word count	476	476	No
dcterms:modified	2013-09-19T10:18:18Z	2013-09-19T10:19:35Z	Yes
meta:creation-date	2013-09-19T10:18:18Z	2013-09-19T10:19:35Z	Yes
meta:save-date	2013-09-19T10:18:18Z	2013-09-19T10:19:35Z	Yes
dc:creator	Kuppuudaiyar, Perumal	Kuppuudaiyar, Perumal	No
Last-Modified	2013-09-19T10:18:18Z	2013-09-19T10:19:35Z	Yes
Author	Kuppuudaiyar, Perumal	Kuppuudaiyar, Perumal	No
dcterms:created	2013-09-19T10:18:18Z	2013-09-19T10:19:35Z	Yes
Complex words	81	81	No
date	2013-09-19T10:18:18Z	2013-09-19T10:19:35Z	Yes
modified	2013-09-19T10:18:18Z	2013-09-19T10:19:35Z	Yes
creator	Kuppuudaiyar, Perumal	Kuppuudaiyar, Perumal	No
xmpTPg:NPages	4	4	No
Creation-Date	2013-09-19T10:18:18Z	2013-09-19T10:19:35Z	Yes
meta:author	Kuppuudaiyar, Perumal	Kuppuudaiyar, Perumal	No
created	Thu Sep 19 11:18:18 BST 2013	Thu Sep 19 11:19:35 BST 2013	Yes
producer	Microsoft® Word 2010	Microsoft® Word 2010	No
CheckSum	4dca60f0ef1fed81150560f5edfb44107cd021e0	e4df74b2802e1979a4ba681ea130ce9ecce2e0b8	Yes
Sentence count	51	51	No
xmp:CreatorTool	Microsoft® Word 2010	Microsoft® Word 2010	No
Content-Type	application/pdf	application/pdf	No
Last-Save-Date	2013-09-19T10:18:18Z	2013-09-19T10:19:35Z	Yes



See the output of `diff-pdf --help` for complete list of options.

Some text	Head1	fgsdg	gsgsgs
s	erdff	34636	45764
dfsdf	fasdf	6363	47474524 5

Figure 5-5: PDF Visual Comparison Sample.

C.4 Latex Document Comparator

The objectives of this comparator is

- To input two latex documents
- Analyse and produce differences in xml file.

C.4.1 Implementation

This comparator tool is developed as java wrapper for latexdiff[33]. Latexdiff is a Perl script, which compares two latex files and marks up significant differences between them (i.e. a diff for latest files).

[33] CTAN perl latexdiff: http://www.ctan.org/tex-archive/support/latexdiff

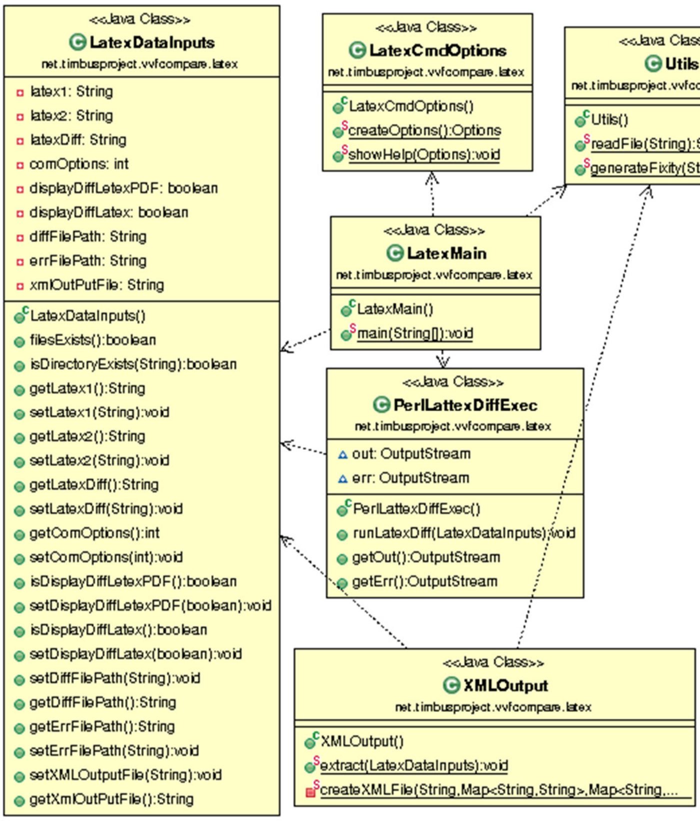
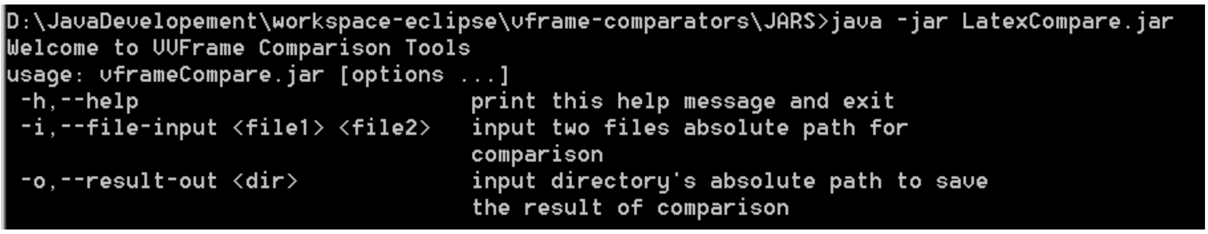


Figure 5-6: Latex Comparator Tool Class Diagram.

C.4.2 Examples

Usage:



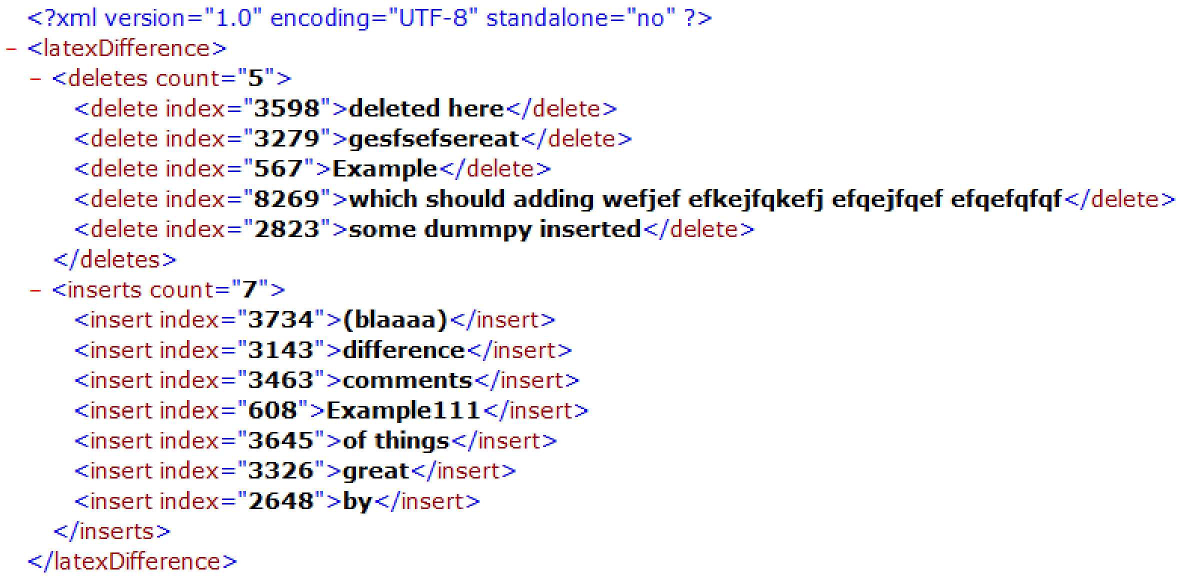
```
D:\JavaDevelopement\workspace-eclipse\vframe-comparators\JARS>java -jar LatexCompare.jar
Welcome to VVFrame Comparison Tools
usage: vframeCompare.jar [options ...]
 -h,--help                      print this help message and exit
 -i,--file-input <file1> <file2>   input two files absolute path for
                                comparison
 -o,--result-out <dir>          input directory's absolute path to save
                                the result of comparison
```

Figure 5-7: Screen capture of Latex Comparator Command line usage.

Script:

```
java -jar LatexCompare.jar -i \sample2e2.tex \sample2e1.tex -o \latex-outputs
```

Output:



```
   <?xml version="1.0" encoding="UTF-8" standalone="no" ?>
 - <latexDifference>
   - <deletes count="5">
       <delete index="3598">deleted here</delete>
       <delete index="3279">gesfsefsereat</delete>
       <delete index="567">Example</delete>
       <delete index="8269">which should adding wefjef efkejfqkefj efqejfqef efqefqfqf</delete>
       <delete index="2823">some dummpy inserted</delete>
     </deletes>
   - <inserts count="7">
       <insert index="3734">(blaaaa)</insert>
       <insert index="3143">difference</insert>
       <insert index="3463">comments</insert>
       <insert index="608">Example111</insert>
       <insert index="3645">of things</insert>
       <insert index="3326">great</insert>
       <insert index="2648">by</insert>
     </inserts>
   </latexDifference>
```

Figure 5-8: Latex Document Compare sample output.

References

(OMG), O. M. (2011). *Business Process Model and Notation (BPMN) Version 2.0.*

Aguilar-Sav, R. S. (2004). Business process modelling: Review and framework. *International Journal of Production Economics*, 129-149.

Alvarez, G., & Petrovi{\'c}, S. (2003). A new taxonomy of web attacks suitable for efficient encoding. *Computers \& Security, 22*(5), 435-449.

Anderson, R. (2001). Why information security is hard - an economic perspective. *Computer Security Applications Conference, 2001. ACSAC 2001. Proceedings 17th Annual*, (pp. 358-365).

Avizienis, A., Laprie, J.-C., Randell, B., & Landwehr, C. (2004). Basic concepts and taxonomy of dependable and secure computing. *Dependable and Secure Computing, IEEE Transactions on, 1*(1), 11-33.

Avizienis, A., Laprie, J.-C., Randell, B., & Landwehr, C. (2004). Basic concepts and taxonomy of dependable and secure computing. *Dependable and Secure Computing, IEEE Transactions on, 1*(1), 11-33.

Basili, V. R., Caldiera, G., & Rombach, H. D. (1994). The Goal Question Metric Approach. In *Encyclopedia of Software Engineering.* Wiley.

Becker, C., Kulovits, H., Rauber, A., & Hofman, H. (2008, June). Plato: a service-oriented decision support system for preservation planning. *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries (JCDL'08).* ACM.

Bellovin, S., Schiller, J., & Kaufman, C. (2003, #dec#). {Security Mechanisms for the Internet}. *{Security Mechanisms for the Internet}(3631)*. IETF.

Bishop, M. (2003). What is computer security? *Security Privacy, IEEE, 1*(1), 67-69.

Blanco, C., Lasheras, J., Fernadez-Medina, E., Valencia-Garcia, R., & Toval, A. (2011, #jun#). Basis for an Integrated Security Ontology According to a Systematic Review of Existing Proposals. *Comput. Stand. Interfaces, 33*(4), 372-388.

Blanco, C., Lasheras, J., Valencia-Garcia, R., Fernandez-Medina, E., Toval, A., & Piattini, M. (2008, March). A Systematic Review and Comparison of Security Ontologies. *Availability, Reliability and Security, 2008. ARES 08. Third International Conference on*, (pp. 813-820).

Blank, R. M., & Gallagher, P. D. (2012, September). *{Guide for Conducting Risk Assessments}.* Tech. rep., National Institute of Standards and Technology, Gaithersburg, MD.

Bowen, P., Hash, J., & Wilson, M. (2006). *SP 800-100. Information Security Handbook: A Guide for Managers.* Gaithersburg, MD, United States: National Institute of Standards \& Technology.

Cavusoglu, H., Mishra, B., & Raghunathan, S. (2004, #jul#). A Model for Evaluating IT Security Investments. *Commun. ACM, 47*(7), 87-92.

Chakrabarti, A., & Manimaran, G. (2002). Internet infrastructure security: A taxonomy. *Network, IEEE, 16*(6), 13-21.

Chung, L., & Prado Leite, J. C. (2009). Conceptual Modeling: Foundations and Applications. In A. T. Borgida, V. K. Chaudhri, P. Giorgini, & E. S. Yu (Eds.). Berlin, Heidelberg: Springer-Verlag.

Commission, E. (2013, Jul). *Cybersecurity Strategy of the European Union: An Open, Safe and Secure Cyberspace.* European Commission Joint Communication, European Commission.

Commission, E. (2013, Jul). Proposal for a Directive of the European Parliament and of the Council concerning measures to ensure a high common level of network and information security across the Union. *2013/0027 (COD)*, 1-20.

Dappert, A., & Farquhar, A. (2009). Significance is in the Eye of the Stakeholder. *Proceedings of the 13th European Conference on Research and Advanced Technology for Digital Libraries* (pp. 297-308). Berlin, Heidelberg: Springer-Verlag.

Denker, G., Kagal, L., Finin, T., Paolucci, M., & Sycara, K. (2003). Security for daml web services: Annotation and matchmaking. In *The Semantic Web-ISWC 2003* (pp. 335-350). Springer.

Donner, M. (2003). Toward a security ontology. *IEEE Security \& Privacy*, 6-7.

Evans, {., Bond, P., & Bement}, A. (2003, Oct). *Guide to Information Technology Security Services.* Special Publication, National Institute of Technology.

Fenz, S. (2010). Ontology-based generation of IT-security metrics. *Proceedings of the 2010 ACM Symposium on Applied Computing*, (pp. 1833-1839).

Fenz, S., & Ekelhart, A. (2009). Formalizing information security knowledge. 183-194.

Fenz, S., Ekelhart, A., & Neubauer, T. (2009, 3). Ontology-based Decision Support for Information Security Risk Management. *International Conference on Systems, 2009. ICONS 2009.* (pp. 80-85). IEEE Computer Society.

Fenz, S., Pruckner, T., & Manutscheri, A. (2009, 4). Ontological Mapping of Information Security Best-Practice Guidelines. *Business Information Systems, 12th International Conference on Business Information Systems, BIS 2009.* Springer Berlin Heidelberg.

Fenz, S., Weippl, E. R., & Ekelhart, A. (2007, 6). Security Ontologies: How to Improve Understanding of Complex Relationships. *Proceedings of the World Conference on Educational Multimedia, Hypermedia and Telecommunications 2007* (pp. 404-407). AACE.

Fenz, S., Weippl, E. R., Klemen, M., & Ekelhart, A. (2007, 1). Security Ontologies: Improving Quantitative Risk Analysis. *Proceedings of the 40th Hawaii International Conference on System Sciences, HICSS2007* (pp. 156-162). IEEE Computer Society.

Firesmith, D. G. (2005). A taxonomy of security-related requirements. *International Workshop on High Assurance Systems (RHAS'05).*

Fraser, B. (1997, #sep#). {Site Security Handbook}. *{Site Security Handbook}(2196)*. IETF.

Grance, T., Hash, J., Stevens, M., O'Neal, K., & Bartol, N. (2003, October). *Guide to Information Technology Security Services.* Tech. rep., National Institute of Technology.

Gruber, T. R. (1995). Toward principles for the design of ontologies used for knowledge sharing? *International journal of human-computer studies, 43*(5), 907-928.

Guttenbrunner, M., & Rauber, A. (2012, 3). A Measurement Framework for Evaluating Emulators for Digital Preservation. *ACM Transactions on Information Systems (TOIS), 30*(2).

Guttenbrunner, M., & Rauber, A. (2012, October 1-5). Evaluating an Emulation Environment: Automation and Significant Key Characteristics. *Proceedings of the 9th International Conference on Digital Preservation (iPres 2012)*, (pp. 201-208). Toronto, Canada.

Guttman, E., Leong, L., & Malkin, G. (1999, #feb#). {Users' Security Handbook}. *{Users' Security Handbook}(2504)*. IETF.

Haren, V. a. (2012). *ArchiMate 2. 0 Specification.* Van Haren Publishing.

Harris, S. (2012, 10). *CISSP All-in-One Exam Guide, Fifth Edition* (6 ed.). McGraw-Hill Osborne Media.

Herzog, A., Shahmehri, N., & Duma, C. (2007). An ontology of information security. *International Journal of Information Security and Privacy (IJISP), 1*(4), 1-23.

Housley, R. (1993, #may#). {Security Label Framework for the Internet}. *{Security Label Framework for the Internet}(1457)*. IETF.

IEEE Std 1012 - 2004 IEEE Standard for Software Verification and Validation. (2005). *IEEE Std 1012 - 2004 IEEE Standard for Software Verification and Validation*, 0_1--110.

Irvine, C., & Levin, T. (1999). Toward a taxonomy and costing method for security services. *Computer Security Applications Conference, 1999.(ACSAC'99) Proceedings. 15th Annual*, (pp. 183-188).

ISO. (2008, #jun#). *{ISO/IEC 27005 Information technology - Security Techniques - Information security risk management}.* ISO. ISO/IEC.

ISO. (2012, Dec). *ISO/IEC 27000:2012 Information technology -- Security techniques -- Information security management systems -- Overview and vocabulary.* Tech. rep., ISO.

ISO/IEC 12207:2008: Systems and software engineering - Software life cycle processes. (2008, #feb#). *ISO/IEC 12207:2008: Systems and software engineering - Software life cycle processes*.

ISO/IEC. (2010). *{ISO/IEC 25010 - Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models}.* ISO. ISO/IEC.

Karyda, M., Balopoulos, T., Dritsas, S., Gymnopoulos, L., Kokolakis, S., Lambrinoudakis, C., et al. (2006). An ontology for secure e-government applications. *Availability, Reliability and Security, 2006. ARES 2006. The First International Conference on*, (pp. 5--pp).

Kaufman, C., Perlman, R., & Speciner, M. (2002). *Network security: private communication in a public world.* Prentice Hall Press.

Kent, S., & Seo, K. (2005, #dec#). {Security Architecture for the Internet Protocol}. *{Security Architecture for the Internet Protocol}(4301)*. IETF.

Kim, A., Luo, J., & Kang, M. (2005). *Security ontology for annotating resources.* Springer.

Kim, G. H., & Spafford, E. H. (1994). The Design and Implementation of Tripwire: A File System Integrity Checker. *Proceedings of the 2Nd ACM Conference on Computer and Communications Security* (pp. 18-29). New York, NY, USA: ACM.

Kissel, R. (2013, May). *{Glossary of Key Information Security Terms}.* Tech. rep., National Institute of Standards and Technology, Gaithersburg, MD.

Lasheras, J., Valencia-Garcia, R., Fernandez-Breis, J. T., & Toval, A. (2009). Modelling reusable security requirements based on an ontology framework. *Journal of Research and Practice in Information Technology, 41*(2), 119.

Martimiano, A., & Moreira, E. (2005). An owl-based security incident ontology. *Proceedings of the Eighth International Protege Conference*, (pp. 43-44).

Miksa, T., Mayer, R., & Rauber, A. (2013). Ensuring Sustainability of Web Services Dependent Processes. *International Journal of Computational Science and Engineering (IJCSE)*.

Miksa, T., Vieira, R., Rauber, A., Proell, S., Strodl, S., & Barateiro, J. (2013, 9). Framework for Verification of Preserved and Redeployed Processes. *Proceedings of the 10th International Conference on Digital Preservation (iPRES2013).*

Mouratidis, H., Giorgini, P., & Manson, G. (2003). An ontology for modelling security: The tropos approach. *Knowledge-Based Intelligent Information and Engineering Systems*, (pp. 1387-1394).

Münch, I. (2008). IT-Grundschutz-Kataloge 2007. *Datenschutz und Datensicherheit, 32*(3), 215.

Nadeem, A., & Javed, M. Y. (2005). A performance comparison of data encryption algorithms. *Information and communication technologies, 2005. ICICT 2005. First international conference on*, (pp. 84-89).

Noy, N. F., McGuinness, D. L., & others. (2001). Ontology development 101: A guide to creating your first ontology. *Ontology development 101: A guide to creating your first ontology*. Stanford knowledge systems laboratory technical report KSL-01-05 and Stanford medical informatics technical report SMI-2001-0880.

P. Missier, S. S. (2010). Janus: From workflows to semantic provenance and linked open data. *Proceedings of the International Provenance and Annotation Workshop (IPAW2010)*, (pp. 129-141). New York, USA.

Paar, C., & Pelzl, J. (2010). *Understanding cryptography: a textbook for students and practitioners.* Springer.

Pethia, R., Crocker, S., & Fraser, B. (1991, #nov#). {Guidelines for the Secure Operation of the Internet}. *{Guidelines for the Secure Operation of the Internet}(1281)*. IETF.

Pohl, K. (2010). *Requirements Engineering: Fundamentals, Principles, and Techniques* (1st ed.). Springer Publishing Company, Incorporated.

Raskin, V., Hempelmann, C. F., Triezenberg, K. E., & Nirenburg, S. (2001). Ontology in information security: a useful theoretical foundation and methodological tool. *Proceedings of the 2001 workshop on New security paradigms*, (pp. 53-59).

Raskin, V., Hempelmann, C. F., Triezenberg, K. E., & Nirenburg, S. (2001). Ontology in Information Security: A Useful Theoretical Foundation and Methodological Tool. *Proceedings of the 2001 Workshop on New Security Paradigms* (pp. 53-59). New York, NY, USA: ACM.

Savola, R. M. (2007). Towards a taxonomy for information security metrics. *Proceedings of the 2007 ACM workshop on Quality of protection*, (pp. 28-30).

Savolainen, P., Niemela, E., & Savola, R. (2007). A Taxonomy of Information Security for Service-Centric Systems. *Software Engineering and Advanced Applications, 2007. 33rd EUROMICRO Conference on*, (pp. 5-12).

Scarfone, K., Souppaya, M., Cody, A., & Orebaugh, A. (2008, September). *{Technical Guide to Information Security Testing and Assessment }.* Tech. rep., National Institute of Standards and Technology, Gaithersburg, MD.

Scarfone, K., Souppaya, M., Cody, A., & Orebaugh, A. (2008, September). *{Technical Guide to Information Security Testing and Assessment }.* Tech. rep., National Institute of Standards and Technology, Gaithersburg, MD.

Shirey, R. (2000, #may#). {Internet Security Glossary}. *{Internet Security Glossary}(2828)*. IETF.

Shirey, R. (2007, #aug#). {Internet Security Glossary, Version 2}. *{Internet Security Glossary, Version 2}(4949)*. IETF.

Simmonds, A., Sandilands, P., & van Ekert, L. (2004). An ontology for network security attacks. In *Applied Computing* (pp. 317-323). Springer.

Simon, F. S. (2010). *Qualitäts-Risiko-Management: Ganzheitliche Projektsteuerung.* Berlin.

Siponen, M. T., & Oinas-Kukkonen, H. (2007, #feb#). A Review of Information Security Issues and Respective Research Contributions. *SIGMIS Database, 38*(1), 60-80.

Souag, A., Salinesi, C., & Comyn-Wattiau, I. (2012). Ontologies for Security Requirements: A Literature Survey and Classification. *Advanced Information Systems Engineering Workshops*, (pp. 61-69).

Stoneburner, G. (2001, December). *{Underlying Technical Models for Information Technology Security}.* Tech. rep., National Institute of Standards and Technology, Gaithersburg, MD.

Storer, M. W., Greenan, K., & Miller, E. L. (2006). Long-term threats to secure archives. *Proceedings of the second ACM workshop on Storage security and survivability*, (pp. 9-16).

Susanne Glissman, J. S. (2009). *A Comparative Review of Business Architecture.* IBM Research.

TIMBUS Consortium. (2012). *D8.1: Use Case Definition and Digital Preservation Requirements.*

TIMBUS Consortium. (2013a). *D4.3: Dependency Models Iter. 2.*

TIMBUS Consortium. (2013b). *D4.6: Use Case Specific DP & Holistic Escrow.*

TIMBUS Consortium. (2013c). *D7.7: Preservation of an Open Source Workflow – Case Description and Analysis.*

Tracy, K. (2008). Identity management systems. *Potentials, IEEE, 27*(6), 34-37.

Tsoumas, B., & Gritzalis, D. (2006). Towards an ontology-based security management. *Advanced Information Networking and Applications, 2006. AINA 2006. 20th International Conference on*, *1*, pp. 985-992.

Tsoumas, B., Dritsas, S., & Gritzalis, D. (2005). An Ontology-Based Approach to Information Systems Security Management. In V. Gorodetsky, I. Kotenko, & V. Skormin (Eds.), *Computer Network Security* (Vol. 3685, pp. 151-164). Springer Berlin Heidelberg.

van Lamsweerde, A. (2009, #mar#). *{Requirements Engineering: From System Goals to UML Models to Software Specifications}.* Wiley.

Venter, H., & Eloff, J. H. (2003). A taxonomy for information security technologies. *Computers \& Security, 22*(4), 299-307.

Wilson, A. (2007). *Work Package 2.2: Significant Properties Report.* Tech. rep., InSPECT.

Y. L. Simmhan, B. P. (2005). A survey of data provenance in e-science. *SIGMOD Rec.*, 31-36.

Yamamoto, S., Williams, C., Yokota, H., & Parent, F. (2009, #aug#). {Softwire Security Analysis and Requirements}. *{Softwire Security Analysis and Requirements}(5619)*. IETF.

Young, R. R. (2004). *The Requirements Engineering Handbook.*

Yu, E. S. (1997). Towards Modeling and Reasoning Support for Early-Phase Requirements Engineering. *Proceedings of the 3rd IEEE International Symposium on Requirements Engineering* (pp. 226--). Washington, DC, USA: IEEE Computer Society.

Zhou, J., & Gollman, D. (1996). A fair non-repudiation protocol. *Security and Privacy, 1996. Proceedings., 1996 IEEE Symposium on*, (pp. 55-61).

Zhou, J., & Gollmann, D. (1997). Evidence and non-repudiation. *Journal of Network and Computer Applications, 20*(3), 267-281.