D4.5: Business Process Contexts

WP 4 – Processes and Methods

for Digitally Preserving Business Processes

Delivery Date: 30/03/2012

Dissemination Level: Restricted

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**
Hedda R. Schmidtke	KIT	schmidtke@teco.edu
Martin A. Neumann	KIT	mneumann@teco.edu
Goncalo Antunes	INESC-ID	goncalo.antunes@ist.utl.pt

Contributors		
Name	**Organisation**	**e-mail**
John Thomson	CMS	john.thomson@caixamagica.pt
Mike Nolan	Intel	michael.nolan@intel.com
Gregor Heinrich	iPharro	g.heinrich@ipharro.com
Martin Hecheltjen	ITM	martin.hecheltjen@uni-muenster.de
Thomas Molka	SAP	thomas.molka@sap.com
Rudolf Mayer	SBA	rmayer@sba-research.org
Daniel Draws	SQS	daniel.draws@sqs.de

Internal Reviewer		
Name	**Organisation**	**e-mail**
Angela Dappert	DPC	angela@dpconline.org
José Barateiro	LNEC	jbarateiro@lnec.pt

Document History			
Version	**Date**	**Author**	**Changes**
V0.1	01/11/2011	All partners	Initial draft
V1.0	24/02/2012	All partners	Second draft
V1.1	07/03/2012	All partners	Draft for 1st review
V1.2	20/03/2012	All partners	Draft for 2nd review
V1.3	30/03/2012	All partners	Release for EU review

Disclaimer

Table of Contents

List of Figures

List of Tables

List of Acronyms

ADM	Architecture Development Method
ARFF	Attribute-Relation File Format
BMM	Business Motivation Model
BPMN	Business Process Model and Notation
CC	Critical Capability
CCR	Controlled Country Regions
CMS	Caixa Magica Software
CPU	Central Processing Unit
CRM	Customer Relationship Management
CUDF	Common Upgradeability Description Format
DCMI	Dublin Core Metadata Initiative
DL	Description Logics
DP	Digital Preservation
DUDF	Distribution Upgradeability Description Format
EIP	Enterprise Information Portal
EPC	Event-driven Process Chain
ERM	Enterprise Risk Management
FOSS	Free and Open Source Software
GNU	"GNU's Not Unix!"
HD	Hard Disk
IDF	Invention Disclosure Form
III-RM	Integrated Information Infrastructure Reference Model
iLE	Intel Labs Europe
INESC-ID	INSTITUTO DE ENGENHARIA DE SISTEMAS E COMPUTADORES, INVESTIGACAO E DESENVOLVIMENTO
Intel	Intel Performance Learning Solutions
IoS	Internet of Services
IP	Intellectual Property

iPharro	iPharro Media GmbH
IT	Information Technology
IT-CMF	IT Capability Maturity Framework
ITM	Westfaelische Wilhelms-Universtitaet Muenster
IVI	Innovation Value Institute
JPG	Joint Photographic Experts Group
KIT	Karlsruher Institut fuer Technologie
OCLC	Online Computer Library Center
OLA	Operation level agreement
OMG	Open Management Group
OS	Operating System
OWL	Web Ontology Language
PGP	Pretty Good Privacy
QRM	Quality risk management
RAM	Random Access Memory
RDF	Resource Description Framework
RDFS	RDF Schema
RLG	Research Libraries Group
SaaS	Software as a Service
SAP	SAP AG
SAS	Statements on Auditing Standards
SBA	Secure Business Austria
SLA	Service Level Agreement
SNOMED	Systematized Nomenclature of Human and Veterinary Medicine
SOM	Self-Organizing Map
SQS	Software Quality Systems
SWRL	Semantic Web Rule Language
TCO	Total Cost of Ownership
TIFF	Tagged Image File Format

TOGAF	The Open Group Architecture Framework
TXT	Trusted Execution Technology
UPC	Underpinning Contracts
URI	Uniform Resource Identifier
VM	Virtual Machine
VRDF	Virtual Resource Description Framework
WP	Work Package
WS-BPEL	Web Service Business Process Execution Language
XML	Extensible Markup Language
XSD	XML Schema Definition

1 Executive Summary

The goal of the TIMBUS project is the preservation of business processes and services. The primary motivation for this is to provide business continuity in face of obsolescing business and computing environments. The presence of Software as a Service (SaaS) and Internet of Services (IoS) means business processes are increasingly supported by service-oriented systems where numerous services provided by different providers located in different geographical locations are composed to form service systems which will continue evolving. Besides the advantages of SaaS and IoS, there is the risk of services and service providers disappearing (for various reasons) leaving partially complete business processes.

Successful digital preservation of business processes requires capturing sufficient detail of a business process and its context to be able to exhume and validate its original behaviour to a level of detail satisfying preservation requirements at a future date under changed and evolved conditions. These include potentially different parties, different enabling technologies, different system components (hardware and software), changed services by different service providers and differences in other aspects of the context of the business process. Digital preservation of business processes and services therefore requires preserving the set of activities, processes and tools, which all together ensure continued access to the services and software which are necessary to reproduce the context within which information can be accessed, properly rendered, validated and transformed into knowledge.

Traditional digital preservation approaches focus on preserving digital objects and their context. The context is in the form of Representation Information, which is the information needed so that certain Designated Communities can understand the digital object in the future, and Preservation Description Information, which is the additional preservation metadata needed to manage the preservation of the digital object (Consultative Committee for Space Data Systems, 2002). From the TIMBUS perspective, the context of preserving processes and services is considerably more complex, since it not only requires dealing with the structural properties of information, but also with the dynamic behaviour of business processes.

In both preservation scenarios it is not clear at a given time, which information will be required later to restore the system. What is a clear boundary of technical feasibility today can be a parameter not accounted for in the future. It is, therefore, important to determine which context parameters very likely need to be preserved by most users in most scenarios, to enable users to identify the context parameters that are important for their specific preservation task, and to enable users to extend the context framework for specialist scenarios.

This document therefore illustrates the relevant aspects for digital preservation of business processes, and it provides a formal meta-model that can be instantiated to capture the relevant aspects of a business process in process-specific models. This modelling approach is mandatory for future modelling activities in TIMBUS. In TIMBUS, a business process is thought of being “in focus” of all digital preservation efforts, and all the aspects and elements that surround a business process constitute the process’ “context” which contains all aspects relevant for a process’ preservation. These relevant aspects surrounding a business process are called “context parameters”.

But the general challenge which is faced in TIMBUS, is the large space of potentially relevant aspects in the context of a process. On the one hand, the set of relevant aspects on the same level of granularity seems to be infinite. On the other hand, the potential levels of granularity seem to be infinite too. Therefore, to scope and to structure the exploration of relevant context parameters, the investigation has been grounded in a divide-and-conquer approach.

To guide the top-down perspective, related work on established enterprise frameworks has been analysed. These models provide holistic but abstract views on the relevant concerns of an enterprise – views which incorporate business processes and relevant aspects from an enterprise perspective. Our analysis has selected the Zachman framework to be most suitable, as it covers business processes and related relevant aspects from various different perspectives which focus on different but distinct concerns, as will be illustrated later on. This way our top-down efforts could be thematically partitioned.

To guide the bottom-up perspective, work on modelling approaches that are related to TIMBUS, as for example the PREMIS data dictionary or CUDF, and other works that serve as a source for digital preservation-relevant aspects have been investigated. With this background in mind, partner-specific scenarios for business process preservation have been designed and their context parameters have been identified.

Both perspectives have been integrated in an ontology-based modelling approach to provide standardized syntax, formal semantics, and sound and complete reasoning mechanisms. In addition, the ontology-based model is motivated and reasoning capabilities are explored by way of example. The model and its reasoning capabilities provide a solid foundation for further TIMBUS efforts, to support and to solve reasoning problems in TIMBUS using state-of-the-art Semantic Technology.

2 Introduction

Digital preservation approaches focus on preserving digital objects and their context. The context is in the form of Representation Information, which is the information needed so that certain Designated Communities can understand the digital object in the future, and Preservation Description Information, which is the additional preservation metadata needed to manage the preservation of the digital object (Consultative Committee for Space Data Systems, 2002). Research, so far has studied the preservation context of files, data sets, software and computer games, with some tentative advances into virtual environment. From the TIMBUS perspective, the context of preserving processes and services is considerably more complex, since it not only requires dealing with the structural properties of information, but also with the dynamic behaviour of business processes.

The overall goal of the TIMBUS project is to enable successful and feasible digital preservation of entire business processes. This is an expansion of the previous approaches that presents completely novel challenges. In TIMBUS, the entire relevant context surrounding the business process itself is to be captured in an automated or semi-automated way, along with the digital objects that are used in the process and their contexts, so that its future exhumation is enabled.

2.1 Context Terminology

The ability to abstract from the context is key to understanding how human beings act efficiently in a very complex world. By separating what is changing and relevant from what is constant or irrelevant, human beings can focus on the task at hand and considerably reduce complexity. Context has been an independent focus of research in the fields of artificial intelligence, cognitive science, and pervasive computing. The aim of this research is on capturing context formally, to construct systems that can deal with a complex world in a very efficient way or that can understand human beings better and thus support them in their everyday activities. The notion of context includes a range of parameters that are generally agreed on. Among these are time, space, actors, and objects.

A definition of context that has been accepted widely in the area of context-aware applications has been given by Dey and Abowd: “Context is any information that can be used to characterise the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.” (Abowd, Dey, Brown, Davies, Smith and Steggles, 1999) A problem with this definition is that it does not differentiate between context and information about context. A more recent definition by Bardram better reflects the fact that context exists outside a representation system: “ ‘Context’ refers to the physical and social situation in which computational devices are embedded.” (Bardram, 2005)

A context model can then be conceived as a model suitable for storing information about the context of an entity in focus. For example, mobile context-aware computing has to cover issues of sensor reliability, ad-hoc network communication, software development support, reasoning and inference, usability, and privacy management. Most recently, ontology-based approaches have gained importance to answer the demands of these heterogeneous application environments with regards to interoperability of context models. The key idea of ontology-based context modelling is that applications using the context model also have to agree on a common ontology, that is, a set of basic concepts defined in a formal language, which developers can use to specify application specific concepts. Application concepts, being founded upon the same basic concepts, can then be used for communication between different applications.

When human beings reason or communicate about entities in the environment, they usually abstract from certain aspects, and reason or communicate within a certain context. Therefore, their reasoning and communication depends on the context the entities are in. Without the dependant context, their reasoning or communication, in worst case, may look absolutely unreasonable or "out-of-context", so to speak. When we reason about space, for instance, we may use "to the West of" as a transitive relation. This assumption is valid as long as we suppose a sufficiently small local area of context, as "to the West of" is globally a cyclic relation: Denmark is "to the West of" Korea, Korea is "to the West of" Canada, and Canada is "to the West of" Denmark; within the local context of a city or country, in contrast, "to the West of" can be used in the same manner as "to the North of", i.e. as if it was a transitive, acyclic relation.

In general, it cannot be explicitly specified what constitutes a business process: it might be a formal syntactic specification (which yields formal semantics), an informal plain text description or it might be merely based on explicit (or even only tacit) knowledge of all people involved in a business process. In consequence, in TIMBUS, an abstract concept of "business process or service" is in focus of any considerations regarding the digital preservation of a business process or service. Furthermore, the terminology "context of a business process or service" refers to the situation of an instance of the abstract concept of a business process or service. Analogously, the terminology "context information of a business process or service" refers to any information that can be used to characterise the situation of an instance of the abstract concept of a business process or service. For example, in a process supported by a software system, the relevant context could consist of the time zone in which the process is executed and the platform that runs the software. Analogously, the relevant context information in this example would be information on the time zone in which the process is executed and information on the platform that runs the software.

2.1 Problem Statement

Digital preservation is traditionally understood as the management of digital information over time. It is the set of processes and activities that ensure continued access to assets existing in digital formats. The digital preservation problem is well-understood for data-centric information scenarios but has been less explored for scenarios where the important digital information to be preserved includes the execution context within which data is processed, analysed, transformed and rendered. Furthermore, preservation is often considered as a set of activities carried out in the isolation of a single domain, without considering the dependencies on third-party services, information and capabilities that will be necessary to validate digital information in a future usage context.

A primary motivation for TIMBUS is the declining popularity of centralized in-house business processes maintained and owned by single entities. The presence of Software as a Service (SaaS) and Internet of

Services (IoS) means that business processes increasingly supported by service-oriented systems where numerous services provided by different providers, located in different geographical locations are composed to form value added service compositions and service systems which will continue changing and evolving. Besides the advantages of SaaS and IoS there is the danger of services and service providers disappearing (for various reasons) leaving only partially complete business processes.

Therefore, successful DP of business processes requires capturing sufficient detail of a business process and its context to be able to re-enable its original behaviour at a future date, involving potentially different participating parties, different enabling technologies, different system components (hardware and software), changed services by different service providers (e.g. IoS or SaaS services) or differences in other aspects of the context of the business process.

In contrast to the preservation of static digital objects where only the ability to render and understand the digital object is the main concern, preservation of dynamic digital objects, such as the preservation of software, computer games and, particularly entire business processes requires the capture of more sophisticated context information. For instance, what is a clear boundary of technical feasibility today can be a parameter not accounted for in the future. Accounting for possible risks through risk management, allows us to identify potential at-risk context information. A brief example is computing speed. Consider a DP time t0, where we have a network connection whose speed is clearly below the speed of a database lookup. Furthermore, we assume there is a race condition: our service internally sends a request to a remote server and concurrently, at the same time looks up a required parameter in the database. As the database access is always faster than the network request, we obtain the parameter before the answer from the remote server arrives. If we exhume the digitally preserved system at a time t1 in a virtual environment, we might get a system in which the simulated network is faster than the database access and the results might not be the same that the original system would have produced. Capturing such constraints is therefore crucial to prevent failure in exhumation of a business process. However, we often do not understand the parameters that make up these constraints. The context in which things happen is usually implicit and natural to the actors at time t0. It is something that they take as a constant, given information to which they do not need to pay attention. It becomes a relevant parameter that has to be modelled only for the actors at time t1, to whom this information is no longer constant and given.

In contrast to digital preservation in other domains, DP of business processes and services has specific requirements. In services and businesses, very often processes are only partially or, even, not at all formalised and change due to highly volatile business environments (Moitra and Ganesh, 2005). From this follows an increased need for autonomy in TIMBUS. DP of business processes and services has to happen automatically or semi-automatically to cope with the volatility. Changes in parts of a system that are not formalised but subject to DP by an agreement between parties, need to be detected without infringing on the privacy rights of parties (e.g. business secrets or personal rights of employees). If such a relevant change is detected by the system, then the system can either trigger a notification (partially autonomous DP) to both parties or a third party, which will perform the DP, or it immediately performs a DP process (completely autonomous DP) itself.

Consequently, we have a double need for capturing context: first, during the expediency phase, for detecting when DP is required (triggering the execution phase and preservation planning), and second, for storing digital contents in context (advising preservation characterisation during the execution phase), so that they can be successfully exhumed later (using appropriate preservation actions). However, in order to know what aspects of the context of a business process to capture in a particular digital preservation case, we need to model its relevant context first. To foster aforementioned autonomy in DP, the context has to be modelled in a way that it can be syntactically and semantically precisely understood by machines.

2.2 Goals

TIMBUS endeavours to enlarge the understanding of DP to include the set of activities, processes and tools that ensure continued access to services and software necessary to produce the context within which information can be accessed, properly rendered, validated and transformed into contextualized knowledge.

The goal of this deliverable is to derive a context model which is sufficiently comprehensive in order to address the different parameters (also called aspects or dimensions) of context of a business process which are relevant from a digital preservation perspective. The context model provides means for capturing the relevant context parameters and contextual dependencies of a business process, that is, the relevant aspects of the situation (and dependencies between these aspects) in which a business process is established. Furthermore, the context model provides the foundation for automated answering of questions (relevant to digital preservation of business processes) by contextual reasoning on context parameters and dependencies.

The deliverable has the following goals:

- Provide a comprehensive **survey** on context parameters of business processes which are relevant from TIMBUS' digital preservation perspective. The context parameters are to be classified according to a well-known and established or standard enterprise framework. This provides a holistic view on the context of business processes of an organization. In this context, the semantics of relevant context parameters are to be informally and formally surveyed to provide insight into their meaning and to allow their formal integration into the context model.
- Design **the context model** that provides a framework to syntactically and semantically model the relevant context parameters and dependencies between them.
    - Deliverables D4.2 and D4.3 focus on the concrete considerations of the (informal and formal) semantics of **dependencies** between the relevant context parameters. They are, therefore, out of scope for this deliverable. When, in this document, we say "context parameters and dependencies between them" then the emphasis of research is on the context parameters themselves and to a much lesser degree on the dependencies, which will be described in the other deliverables. It is, however, important not to ignore them since the deliverables have to seamlessly fit together.

  - WP6 focuses on the concrete implementation of how to capture the relevant context parameters and the dependencies between them (which may either be based on automated or manual approaches). They are, therefore, out of scope of this deliverable.
  - An important issue to address is which modelling standards to reuse within our context model. Many relevant context parameters and the dependencies between them are (syntactically and semantically) modelled using widely accepted standard modelling techniques. For example, the causal flow of a business process may be modelled using BPMN. It has to be determined how to integrate other standards' semantics (and optionally their syntax) with the context model. This provides interoperability with established standards in modelling of relevant context parameters and the dependencies between them and, therefore, eases their capture in WP6.
- Provide the foundation for the deliverables D4.2 and D4.3.
  - Deliverables D4.2 and D4.3 focus on the concrete considerations of the (informal and formal) semantics of **dependencies** between the relevant context parameters. The context model, therefore, needs to provide a flexible framework for defining the formal semantics of dependencies between relevant context parameters.
  - Deliverables D4.2 and D4.3 will provide a comprehensive survey of visions, goals and examples of what TIMBUS goals can be achieved by which kinds of automated reasoning on the context model, To achieve efficiency in the reasoning process built on top of the context model, the context model needs to provide an adequately restricted framework for defining the formal semantics of dependencies between relevant context parameters.

2.3 Approach

The first goal is addressed by a scenario-motivated survey of relevant context parameters. This survey consists of a list of business process use cases that are relevant for digital preservation from a TIMBUS perspective and which informally point out context parameters and dependencies between them.

To provide structured guidance to this approach, the related work section of this document investigates a set of well-known and established enterprise modelling frameworks. We derive a classification scheme from one of those modelling frameworks, the Zachman framework, which is then used in a second step, to categorize the relevant context parameters identified in these scenarios.

To address the second and third goals, the formal specification of the TIMBUS context model is based on the results of the survey of relevant context parameters. The model presents a syntactically and semantically unified approach to modelling all relevant context parameters and the dependencies between them around the abstract concept of a business process.

2.4 Document Structure

The document starts with a related work survey on well-known and established enterprise modelling frameworks. This forms the foundation for the classification scheme of relevant context parameters which will be used throughout the survey of relevant context parameters. Furthermore, relevant context modelling techniques are presented in the related work. This survey provides an insight to the reader of what concrete modelling capabilities the context model originating from this deliverable has to provide.

Next, a survey of scenarios informally illustrating, motivating, and describing context parameters of business processes which are relevant from TIMBUS' digital preservation perspective is described. Based on these scenarios, a survey on the informal semantics of relevant context parameters is summarized in lists which are structured according to the classification scheme which has been established from enterprise modelling frameworks surveyed in the related work section.

This is followed by the formal specification of the context model. The specification starts out with a motivational section that provides an intuitive introduction to the semantics of the context model to the reader. This is followed by the formal abstract syntax, the formal concrete syntax and the formal semantics of the context model. The syntax provides the necessary means for exchanging instances of context models, while the semantics associate the appropriate meaning to instances of the context model, which forms the essential foundation for reasoning on instances of the context model. For example, to verify the (semantic) correctness of an instance of the context model or to incorporate it in more complex decision making, which is in focus of deliverable D4.2.

2.5 Overview on Context

We can think of a multitude of context parameters surrounding a business process. In fact, it is a difficult exercise trying to imagine which parameters are important or matter for preservation purposes, especially when thinking holistically. When we think of an organization and its business processes, there are several stakeholders involved, from the chairman that sets the mission, vision and strategy, to the technical operator, responsible for operating a system involved in the making of a product. Each of these stakeholders has a different interest in the system, a different way of looking at it, and thus different preservation concerns which might be answered by capturing the context parameters that matter in the point of view of the stakeholder. This insight is formalised in the notion of the “Designated Community” of the OAIS reference model (Consultative Committee for Space Data Systems, 2002) for whom we preserve the digital entity. Each Designated Community requires a different network of “Representation Information” to meet the Designated Community’s requirements for preservation.

From the point of view of stakeholders at the executive management level, the relevant contextual aspects surrounding a process are mainly related to the environment surrounding the process. Relevant aspects might include the business goals to be accomplished, the legal framework applicable to the process, the organisational context surrounding the process, the external stakeholders of the process, among others. From a legal point of view, three different approaches regarding the holistic collection of business context by a digital preservation system should be considered: admissibility (which data can be legally collected and processed by the DP system), need (which context information is needed to legally analyse a situation or a process), and motivation (what could be a legal motivation for business companies to preserve context).

From the point of view of stakeholders at the business management, the relevant context of a business process will involve aspects directly related to the process. Aspects such as events triggering processes, policy and strategy rules for decisions, the behaviour accomplished in the process, the information entities consumed/produced by the process, the associated costs, the locations where the process is being performed, the resources used by a process, the actors that are involved in a process, the roles played by the actors involved, and so on.

From the point of view of stakeholders at the software engineering level, the context of a business process will necessarily involve aspects surrounding the software systems that provide support to the process. Execution aspects such as process execution time, geo-location (especially important given the adaption of off-site IT storage and processing facilities), user access-rights, agents involved, the input data, the results produced, and many more might be relevant for the correct preservation of a business process. Additionally, design aspects such as specifications of the software, the algorithms and heuristics used, other documentation and bug-fixes, among others, are also relevant. In a service-oriented landscape, information on aspects such as availability and throughput, locations in which a service is allowed to run, the relevant priorities of the service versus other services, the time-windows when a service is needed, the classification of the data contained within the service, or even for how long to keep service data after the service is no longer running, might be relevant.

Finally, from the point of view of the stakeholders at the infrastructure/hardware engineering level, the relevant context of a process will involve aspects describing the attributes of the physical landscape which hosts the execution environment. Attributes and capabilities of the underlying hardware infrastructure which supports the execution environment may include information such as the host server capabilities (CPU speed, RAM, disk space, throughput speeds, location, embedded technologies, etc.).

3 Related Work

This chapter covers two types of related works. Firstly, a survey on prominent enterprise modelling frameworks in the literature is presented. This establishes the foundation and background knowledge for section 3.3. That section presents our reasoning to use the Zachman framework in TIMBUS to focus and constrain our top-down efforts in exploration of relevant business process context parameters to aspects relevant to enterprises from various stakeholders' perspectives.

Secondly, modelling approaches for sets of context parameters that are relevant to TIMBUS are presented for their reuse in the "Context Model". "Reuse" in this context means that these approaches' formal semantics either need to be integrated into our final context formalization, or our formalization's syntax and semantics is to be based on them. For this purpose, section 5 covers the parameters identified to be relevant in TIMBUS, and section 6 covers our formalization of them, called the "Context Model", which incorporates these parameters, also incorporating the semantic background knowledge introduced by the related modelling techniques in this section.

3.1 Enterprise Modelling Frameworks

This section presents a survey on prominent enterprise modelling frameworks in the literature. In this deliverable, one of these works will focus and constrain our top-down efforts in exploration of relevant business process context parameters to aspects relevant to enterprises from various stakeholders' perspectives.

3.1.1 Zachman Framework

The Zachman Framework was one of the first enterprise architecture frameworks created. It is considered a means for defining the role of information systems in the enterprise, with the purpose of providing a holistic view of the organisation (Zachman, 1987). Functioning as a "classification theory about the nature of the enterprise", and presenting the "kinds of entities that exist within"[1].

Figure 1 depicts the Zachman framework. It is represented as a table where each cell can be related to a set of models, principles, services, and standards needed to address the concerns of one or more stakeholders. The rows depict the viewpoints of the stakeholders of an organisation on the organisation itself: Scope, which defines the business context, including the business purpose and strategy; Business Model, which describes the organisation; System Model, which describes how the systems will satisfy the organisation's information needs, in a way independent from implementation; Technology Model, which describes the implementation of the systems; Components, which details each of the system's components before production; and Instances, which gives a view of the functioning system in its operational environment.

[1]http://www.zachmaninternational.us/index.php/ea-articles/100-the-zachman-framework-evolution

	DATA What	**FUNCTION** How	**NETWORK** Where	**PEOPLE** Who	**TIME** When	**MOTIVATION** Why
SCOPE (contextual)	List of things important in the business	List of business processes	List of business locations	List of important organizations	List of events	List of business goals and strategies
BUSINESS (conceptual)	Conceptual data/object model	Business process model	Business logistics system	Work flow model	Master schedule	Business plan
SYSTEM (logical)	Logical data model	System architecture model	Distributed systems architecture	Human interface architecture	Processing structure	Business rule model
TECHNOLOGY (physical)	Physical data/class model	Technology design model	Technology architecture	Presentation architecture	Control structure	Rule design
COMPONENTS (detailed)	Data definition	Program	Network architecture	Security architecture	Timing definition	Rule specification
INSTANCES (functioning enterprise)	Usable data	Working function	Usable network	Functioning organization	Implemented schedule	Working strategy

Figure 1: Zachman Framework

The columns express different perspectives on each of the viewpoints: Data/What, refers to the information and data objects of the organization; Function/How, describes the functioning of the organization and its systems; Network/Where, refers to spacial elements and their relationships; People/Who, refers to the actors of the organisation and of its systems; Time/When, refers to timing and events; and Motivation/Why, refers to the overall motivation, rules and constraints to the objectives.

Although it makes suggestions on the types of models/contents that might occupy each of the cells, the Zachman Framework does not make prescriptions. The relationships among cells have to be enforced by the methods and models used together with the framework.

3.1.2 The Open Group Architecture Framework

The Open Group Architecture Framework (TOGAF) is an Enterprise Architecture framework which provides methods and tools to support the architecture development (The Open Group, TOGAF version 9.1. Van Haren Publishing, 2011). The framework is divided into seven parts: (*i*) the Introduction, which provides core concepts and definitions; (*ii*) the Architecture Development Method (ADM), which provides an iterative method for the development and governance of the architecture; (*iii*) the ADM Guidelines and Techniques, supporting the ADM; (*iv*) the Architecture Content Framework, which provides a meta-model and example contents for the architecture; (*v*) the Enterprise Continuum and Tools, which provides a classification scheme for organizing the architecture and reference models; (*vi*) the TOGAF Reference Models, which provides standard reference models and terminology that can be used in the architecture effort; and finally (*vii*) the Architecture Capability Framework, which defines the necessary organizational capabilities in order to put into practice the architecture effort. Of relevance to this deliverable are the Architecture Content

Framework, which defines a meta-model for the enterprise, and the TOGAF Reference Models, which define taxonomies embracing different aspects of an organization. Both can be considered a potential source of context parameters.

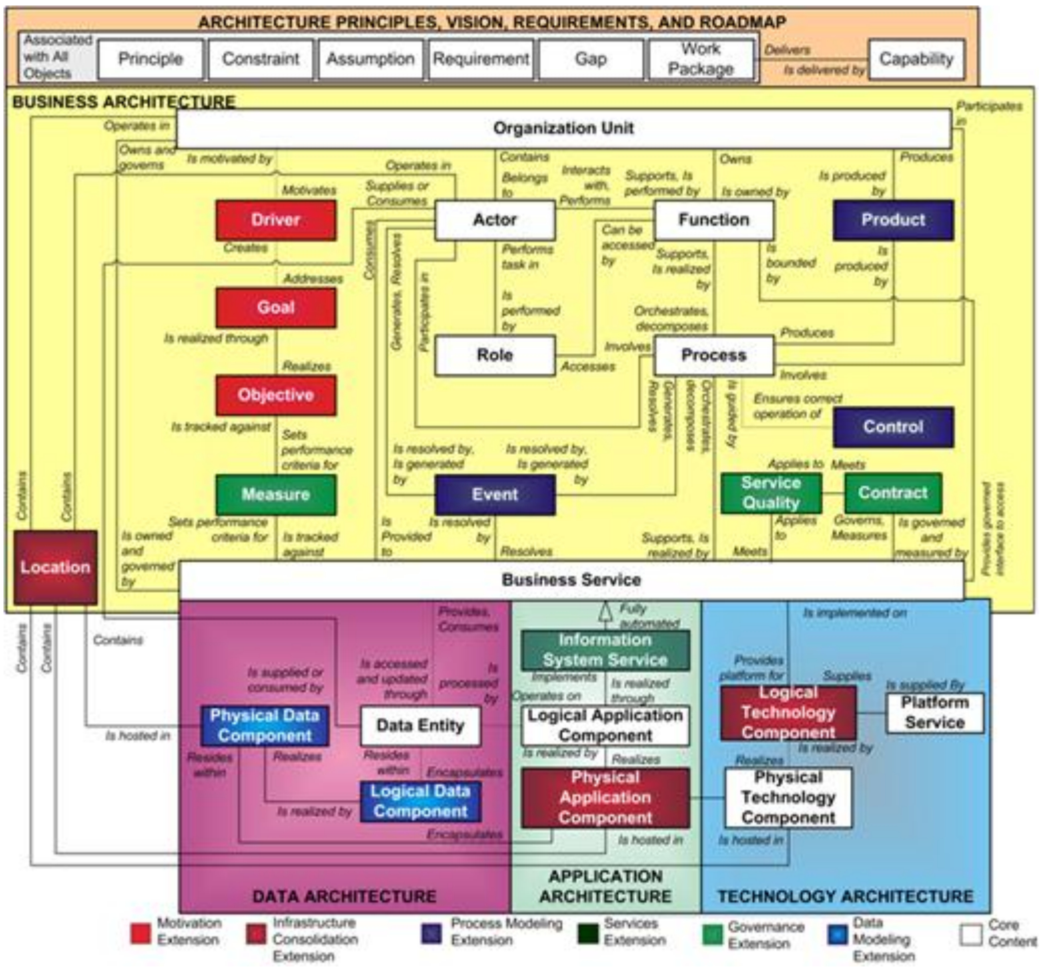


Figure 2: TOGAF Content Meta-model

Source: TOGAF 9.1 Specification

The Architecture Content Framework is TOGAF's alternative to the use of any other available architecture framework (e.g., Zachman framework). The Content Framework defines the possible types of architecture products along with a meta-model. The meta-model provides a definition of all the entities of the architecture that allow architectural concepts to be captured, stored, queried, and represented so that traceability and consistency is achieved between views. Figure 2 depicts the meta-model.

The TOGAF Reference Models comprises two architectural reference models: the TOGAF Foundation Architecture and the Integrated Information Infrastructure Reference Model (III-RM). The TOGAF Foundation Architecture defines a taxonomy for the components and conceptual structure of an information system, defining three types of components (i.e., Applications, Application Platform, and Communications Infrastructure) along with the interfaces between them (i.e., Application Platform Interface and Communications Infrastructure Interface). The III-RM defines a taxonomy and conceptual structure of an integrated information infrastructure that allows the integrated access to information in the organization. The III-RM focuses on Application Software and Application Platform, and on the qualities to which these entities must adhere (e.g., Security, Mobility, Performance SLA's, Management Policies).

3.1.3 ArchiMate

ArchiMate is a modelling language for enterprise architecture (Lankhorst, 2005). It uses an architectural framework for structuring the concepts and relationships of the language, which is divided in three horizontal layers (Business, Application and Technology), which are further refined according to three distinct aspects (Information, Behavior and Structure).

The Business layer addresses the domains of Information, Product, Process and Organization. The Application layer addresses the domains of Application and Data. The Technology layer addresses the domain of the Technological Infrastructure. As for the aspects, the Structure aspects represent the key to interpret the layer. The Behavior aspects describe what the Structure aspects do. The Information aspects comprise the entities that are used by the Behavior aspects. The objective is to model the relationships between the concepts pertaining to different architectures. In that sense, ArchiMate models the context of architecture entities.

Figure 3 depicts the meta-model for the Business layer used in ArchiMate. According to it, a *Business Process* is a *Business Behavior Element* which can be triggered by or might trigger other *Business Behavior Elements*, including other *Business Processes*. Entities that are directly related to *Business Behavior Elements* are the *Business Role* that is assigned to it, the *Business Services* used or realized by it, and the accessed *Business Objects*. Indirect relations with other elements are also easy to determine from the direct relations.

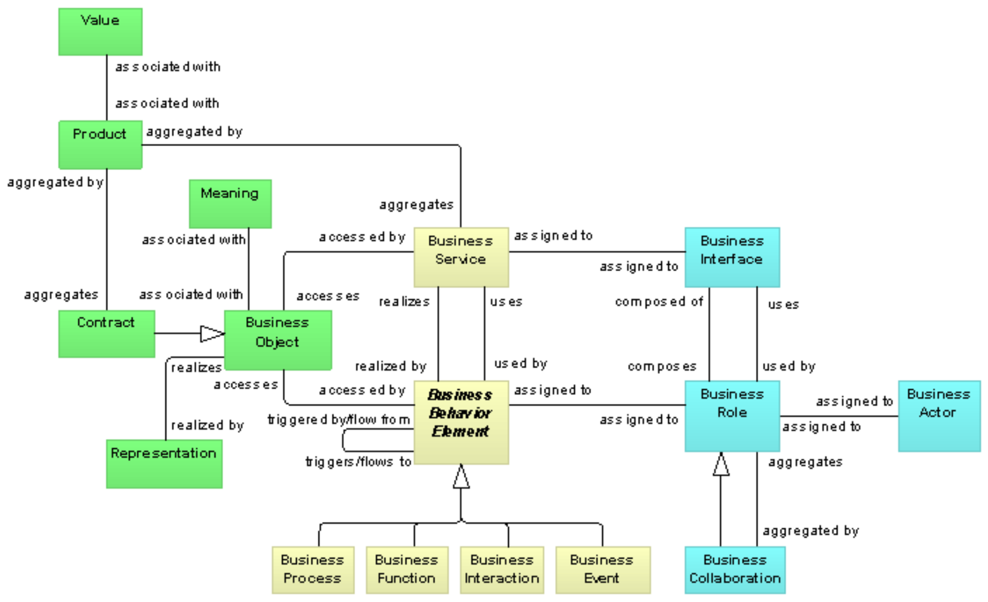


Figure 3: Business Layer Meta-model

Source: Archimate 1.0 Specification

Contextual relationships can also be observed with other layers through cross layer dependency. Figure 4 depicts relationships between the Business layer and the Application layer, where direct relationships between *Business Behavior Element* and entities of the Application layer are depicted.

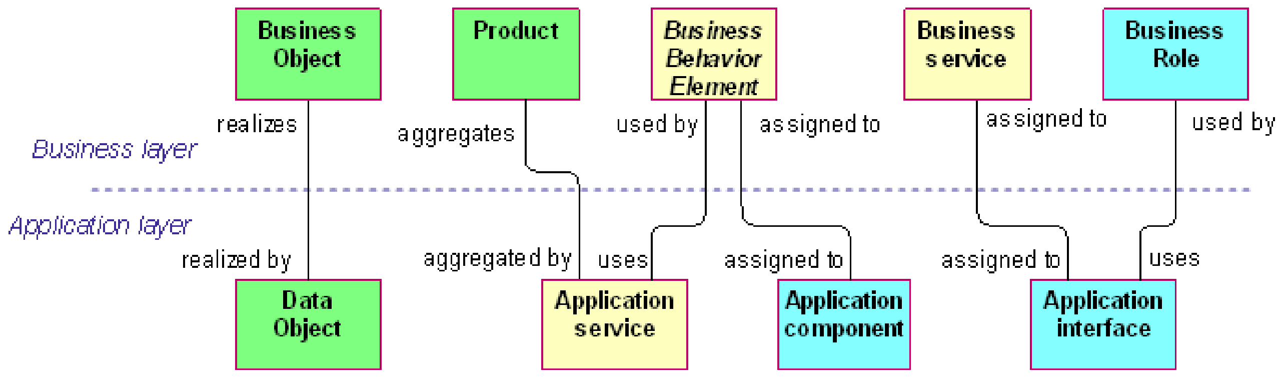


Figure 4: Cross Layer Dependencies between Business and Application Layers

Source: Archimate 1.0 Specification

3.1.4 Business Motivation Model

The Business Motivation Model (BMM), proposed by the Object Management Group (OMG), provides a conceptual framework for the development, communication, and management of business plans, identifying motivational factors behind it, the elements required for its formulation, and relationships between the elements (Object Management Group, 2010). Figure 5 depicts the main relationships between the main concepts of the specification.

Four major concepts are defined: *End*, *Mean*, *Influencer*, and *Assessment*. An *End* is "what an enterprise wants to be". *Ends* can be the *Vision* for the organization (what the organization wants to be); or a *Desired Result*, which can be either a *Goal* (long-term, comprised of *Objectives*) or an *Objective* (short-term, component of *Goals*). A *Mean* is "what an enterprise has decided to do in order to become what it wants to be", in other words, to achieve its *Ends*. *Means* can be the *Mission*, which describes what an organization does in order to achieve the *Vision*; *Courses of Action*, which can be a *Strategy* or *Tactic*; and *Directives*, which govern the *Courses of Action*.

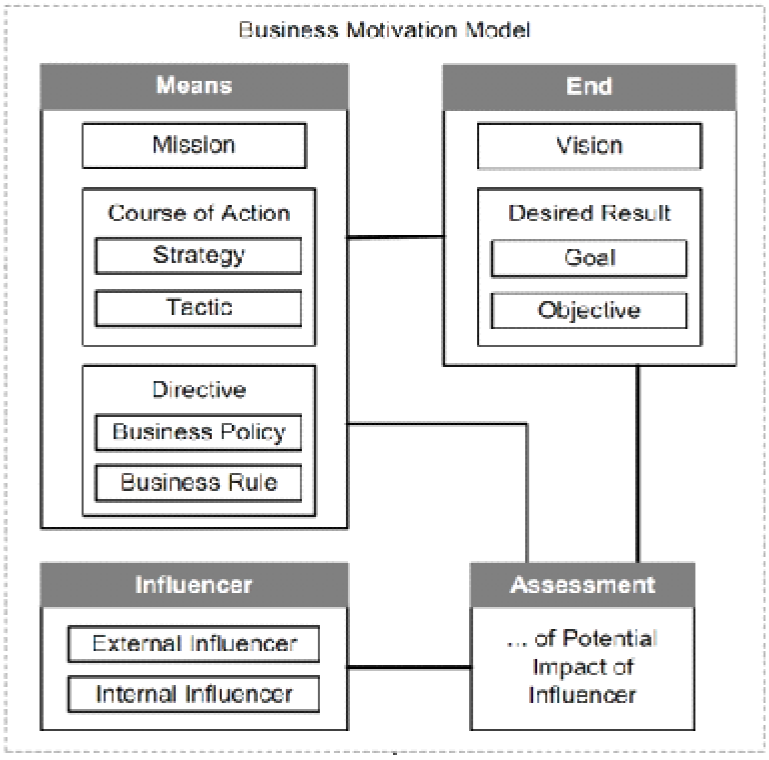


Figure 5: Main Concepts of the Business Motivation Model

Source: BMM Specification

An *Influencer* is "something that can cause changes that affect the enterprise in its employment of its *Means* or achievement of its *Ends*". An *Influencer* can be *External* (from outside the organization; i.e., Applicable Legislation) or *Internal* (from within the organization; i.e., Available Resources). An *Assessment* is "a judgment about the influence" of an *Influencer* "on the enterprise's ability to employ its *Means* or achieve its *Ends*", with decisions stemming from that being reflected in changes to the *Ends* and/or *Means*.

3.1.5 Managing IT like a Business; the IVI IT-CMF Framework

Initially developed by Intel's IT division, the IT capability maturity framework (IT-CMF) is now owned by the Innovation Value Institute (IVI) consortium (http://ivi.nuim.ie/) based in NUI, Maynooth, Ireland. IVI and Intel continue to work closely on joint development of the IT-CMF but now this activity happens under the auspices of the Intel Labs Europe (iLE) rather than Intel IT. IVI's membership spans academic, industry, consulting, analyst, and professional bodies around the world. The basic premise of the IT-CMF is to enable better management and continual development of an organisation’s IT capability to deliver higher business value. More than 200 companies around the world currently use the IT-CMF. It consists of four inter-related strategies for improving IT capability, identifying and prioritising opportunities, reducing costs, and optimising the business value of IT investments. Figure 6 below shows the four main pillars of the IT-CMF.

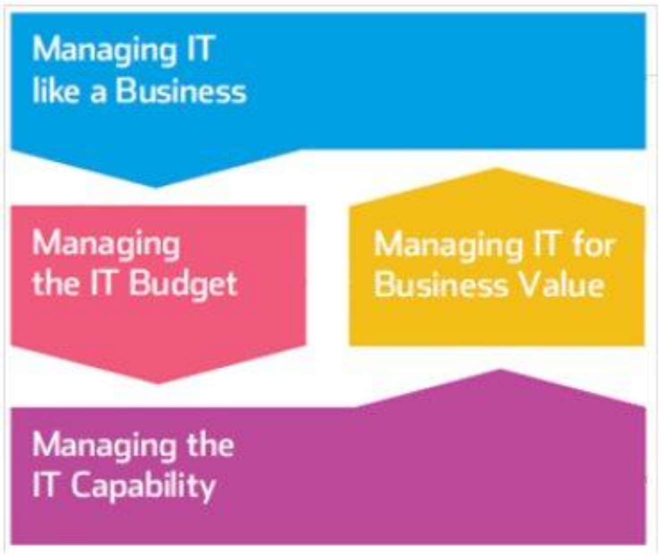

Managing IT like a Business
Managing the IT Budget
Managing IT for Business Value
Managing the IT Capability

Figure 6: The four pillars of the IT-CMF

Source: IVI, 2010

Under each of these pillars, the IT-CMF defines critical capabilities (CC's) as shown below in Figure 7. The IT-CMF is under constant development and it assists organisations by helping to assess their IT practices under the CC categories. This is done by defining 5 maturity levels for each CC and determining which maturity level is closest to your organisations capability which produces an appraisal of the current maturity as well as identification of areas which can be targeted for improvement by taking steps to develop the services required to bring that particular CC to the next maturity level.

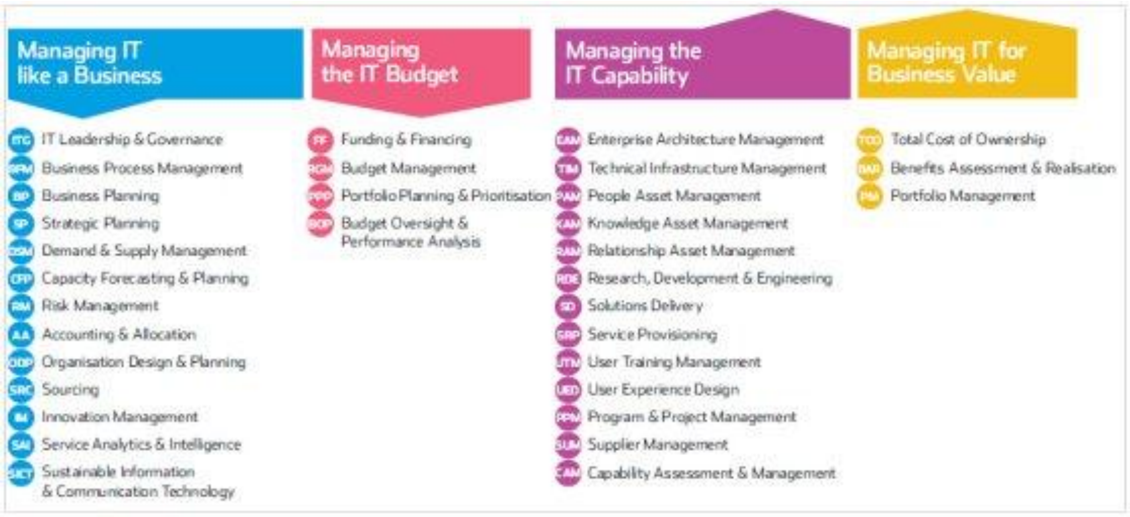

Managing IT like a Business
ITG IT Leadership & Governance
BPM Business Process Management
BP Business Planning
SP Strategic Planning
DSM Demand & Supply Management
CFP Capacity Forecasting & Planning
RM Risk Management
AA Accounting & Allocation
ODP Organisation Design & Planning
SRC Sourcing
IM Innovation Management
SAI Service Analytics & Intelligence
SICT Sustainable Information & Communication Technology
Managing the IT Budget
FF Funding & Financing
BGM Budget Management
PPP Portfolio Planning & Prioritisation
BOP Budget Oversight & Performance Analysis
Managing the IT Capability
EAM Enterprise Architecture Management
TIM Technical Infrastructure Management
PAM People Asset Management
KAM Knowledge Asset Management
RAM Relationship Asset Management
RDE Research, Development & Engineering
SD Solutions Delivery
SRP Service Provisioning
UTM User Training Management
UED User Experience Design
PPM Program & Project Management
SUM Supplier Management
CAM Capability Assessment & Management
Managing IT for Business Value
TCO Total Cost of Ownership
BAR Benefits Assessment & Realisation
PM Portfolio Management

Figure 7: Critical Capabilities of the IT-CMF

Source: IVI 2010

For illustrative purposes, some examples of generic maturity levels are given in the table below. In practice, the capabilities at each level are specifically tailored for each individual CC.

Table 1: Generic IT-CMF Maturity Levels

Source: Deliverable B4c SLA@SOI project

Maturity Level	Capabilities
LEVEL 1 Initial	Management of CCs at this level is ad-hoc & based on individual efforts with no systematic improvement attempts. IT may be viewed somewhat negatively as a necessary expense whose ROI is hard to measure. Budget planning is almost non-existent.
LEVEL 2 Basic	Some effort has gone into understanding the IT landscape. This may be documented informally or in silos. Some tactical-level shared-thinking is beginning to emerge but not on a joined-up, organisational or strategic basis. IT is viewed as a ‘cost centre’ and seen simply as a technology supplier to the business. Focus is on predictable IT service performance and TCO.
LEVEL 3 Intermediate	Formal organisation-wide documented processes are in place to help understand the IT landscape. It is often possible to identify & address gaps. IT is viewed as a ‘service centre’ and a technology expert. There is a systematic approach to cost reduction. ROIs are easier to measure and are based clearly on individual business cases.
LEVEL 4 Advanced	Well established, effective and proven processes exist, which yield a comprehensive picture of the IT landscape. Efficiency is evident; gaps are systematically identified and pro-actively addressed. IT is aligned to business strategies. IT is viewed as an ‘investment centre’. As a strategic business partner, IT engages actively in long-term strategic budget planning to meet the needs of the organisation.
LEVEL 5 Optimising	IT is enabling and influencing future business strategies. Documented IT processes are optimised for efficiency and regularly reviewed. IT is viewed as a ‘value centre’ and a core competency of the organisation.

To perform an assessment, an IT organisation will score each capability from 1-5 according to Table 1. This can be done as a light process which just produces approximate scores as shown below in Figure 8, or it can be a quite detailed report backed up with detailed explanations of the score given along with identification of next steps to improve scores.

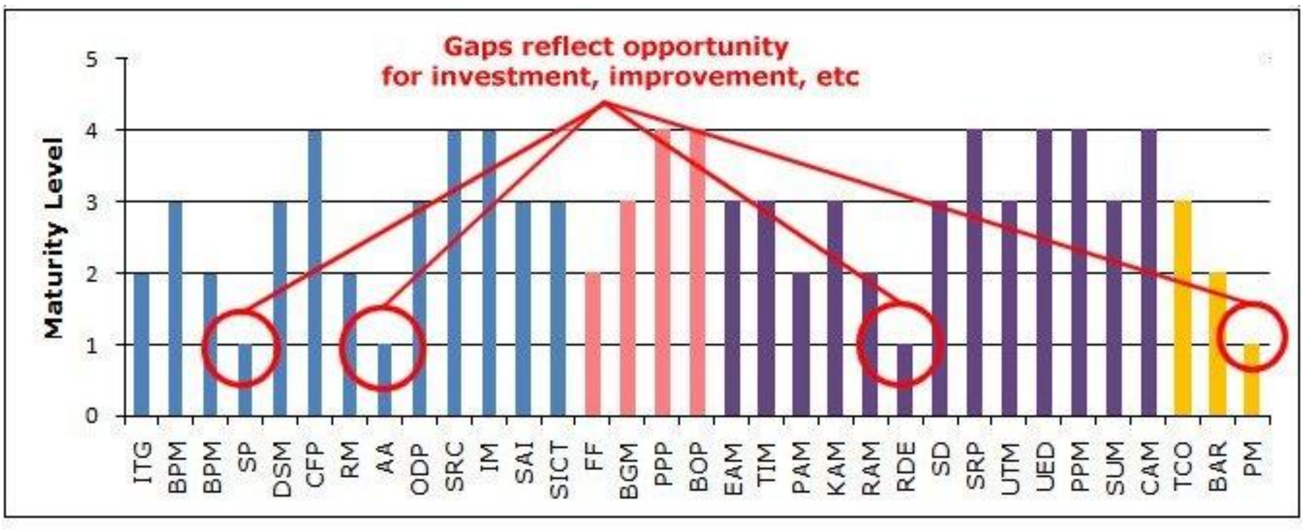


Figure 8: Sample IT-CMF Assessment

Source: Deliverable B4c SLA@SOI project

The IT-CMF is relevant to TIMBUS in several ways. It is under constant development and TIMBUS is collaborating with IVI on two new knowledge management CCs to understand how long-term archival requirements may influence the data management policies or organisations. The CCs also cover areas such as IT architecture, technical infrastructure management, business planning, risk management, solution delivery and cost of ownership which could equally be modified to take into consideration long-term archival requirements. In return, the IT-CMF can also help to structure how the business value will flow from the TIMBUS project in a way which could be related to non-technical decision makers within the business.

The IT-CMF can influence design decisions within an IT organisation. While the IT-CMF does deal with knowledge asset management, there are currently no specific long-term retention considerations supported. It is envisioned that the IT-CMF will grow to incorporate the output of TIMBUS to help inform industry decision by providing structured tiers of capabilities which could be implemented to improve the ability of IT to support the long-term retention needs of the business and shape their preservation strategies.

3.2 Modelling Techniques for Relevant Context Parameters

In this section, modelling approaches for sets of context parameters that are relevant to TIMBUS are presented for their reuse in the “Context Model”. “Reuse” in this context means, that these approaches’ formal semantics either need to be integrated into our final context formalization, or our formalization’s syntax and semantics is be based on them.

3.2.1 Ontologies

In (Staab and Studer, 2009), an ontology is defined as follows: „An ontology is a formal, explicit specification of a shared conceptualization for a domain of interest.“ This definition focuses on the important characteristics of an ontology. An ontology specifies an abstract model of a domain of our world, also referred to as the „domain of discourse“. The model formally defines all concepts and their relationships which are relevant in the domain of discourse.

Depending on the scenario, the complexity of ontologies may vary. The expressiveness of ontologies ranges from basic taxonomies to complex networks of concepts, relationships and rules on these concepts and relationships. The band width of expressiveness of ontologies is reflected by their degree of (mathematical) formalization. Formalization ranges from informal descriptions, over semi-formal specifications to ”fully-fledged“ ontologies, which are based on a formal semantics (usually grounded in mathematical logics) providing sound and complete reasoning capabilities on ontologies. Due to this range in expressiveness and formalisation, ontologies are usually classified into light-weight (restricted expressiveness and formalization) and heavy-weight (high expressiveness and entire formalization) ontologies.

Depending on its features, an ontology may contain the following elements:

- Classes: hierarchically organized concepts relevant in the domain of discourse (taxonomy)
- Instances: relevant class instances in the domain of discourse
- Relations: relations between classes and/or instances
- Axioms: statements on the classes, instances and/or relations
- Rules: conditional statements on the classes, instances and/or relations

An exemplary ontology is shown in Figure 9.

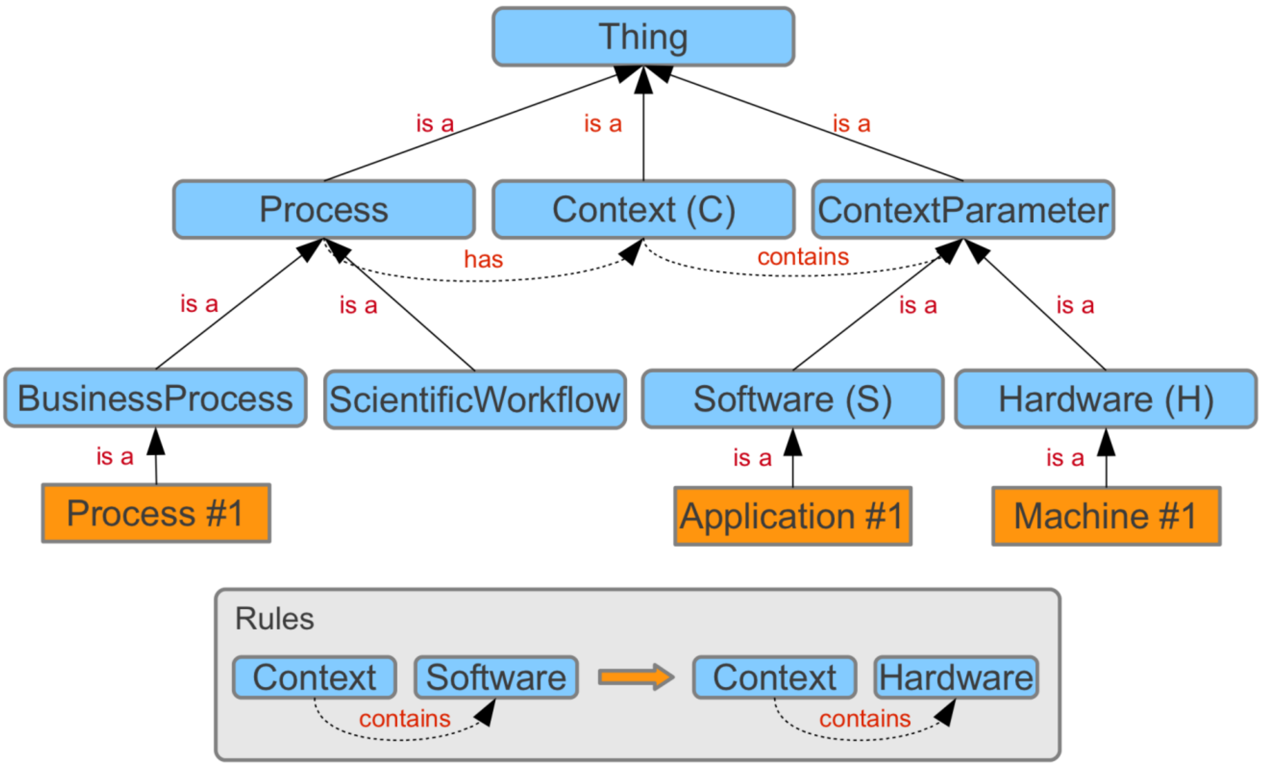


Figure 9: Example Ontology

In particular, heavy-weight ontologies are usually grounded in mathematical logics. Therefore, they can usually be specified using an ontology language which provides a formal syntax. The syntax usually provides the necessary tools to compose complex classes and relations from already defined classes and relations using appropriate operations (also referred to as constructors). In a particular heavy-weight ontology languages the set of constructors usually contains at least the “traditional” logical conjunctives: “and”, “or”, “not” and “existential, cardinal, universal quantification”. As will be illustrated later, concrete ontology languages may provide additional constructors.

In addition, as mentioned before, with heavy-weight ontologies, this formal syntax provides the ability to describe ontologies that have a model-theoretic mathematical semantics associated with them, which in turn provides sound and complete reasoning capabilities on the ontologies. These reasoning capabilities may for example be used to infer implicit knowledge from the explicit knowledge in an ontology, or during knowledge engineering, i.e. the design phase of an ontology, to verify its correctness by proving the non-existence of contradictions.

Relations between classes, as for example the “is a” relation, are one of the fundamental building blocks in ontology reasoning. Using for example the “is a” relation, an ontology reasoner has the capability to abstract and specialize its knowledge on the type of individuals to infer further properties of these individuals which are not part of the explicitly given knowledge. An overview on ontology reasoning functionality based on examples follows later in the section on OWL.

In the following, the topics of expressiveness, formality and reasoning capabilities are discussed based on two of the most prominent ontology languages, namely RDF(S) and OWL.

3.2.1.1 RDF(S)

The Resource Description Framework (RDF) is founded on the idea of formulating basic statements on resources. This is implemented in RDF by providing a formal syntax and semantics for so-called triples. Each triple consists of a “subject”, a “predicate” and an “object”, usually denoted as “(subject predicate object)”. A set of one or more RDF triples is referred to as an “RDF knowledge-base”.

For example, the triple “(MusicProcess isScenarioIn TIMBUS)” means that the resource identified by “MusicProcess” is related to the resource identified by “TIMBUS” using the resource identified by “isScenarioIn”. Informally, this could have the following meaning (e.g. to a knowledge engineer): the music process (in section 4.4) is a relevant scenario in the TIMBUS EU FP7 project. But as the RDF framework does not differentiate between entities and relations, the aforementioned informal meaning, is not its meaning in RDF triples and thus this meaning is not machine-interpretable by a generic RDF engine. One could obviously implement a specific system that associates this meaning to the RDF triples, but in this case the semantics is kept in this dedicated application and not in the RDF triples itself. This issue is addressed by the semantics of RDFS.

Resources can either be Uniform Resource Identifiers (URIs), literals or “blank nodes”. RDF uses URIs to provide the ability to relate resources distributed across remote knowledge-bases. In addition, resources may, for example, be numeric or textual literals, for example, “MusicProcess” or “7”. The available data types in RDF are given by XML Schema Definition[2] (XSD). Finally, "blank nodes" represent anonymous resources, which can informally be interpreted as "existential quantification": the related resource is unknown, but we know that there is at least one.

A set of RDF triples (i.e. an RDF knowledge base) can syntactically be describes by:

- RDF/XML (formal syntax specification)
- A list of RDF triples (informal syntax specification)
- A graph of nodes (subjects and objects) and edges (predicates) (informal syntax specification)

RDF knowledge bases can be queried in an analogous style to relational databases. For example, a users may want to look up all scenarios in the TIMBUS project, by querying the knowledge base with a query triple that uses wildcards: "(* isScenarioIn TIMBUS)". For example, SPARQL[3] is a prominent querying language for RDF knowledge bases.

As mentioned before, all three elements of an RDF triple "(subject predicate object)", i.e. subject, object and predicate, are "Resources" in RDF. This is a very special notion in RDF. Informally, subject and object "Resources" might be interpreted by external users as entities, and predicate "Resources" might be interpreted as relations, but in RDF syntax and semantics these concepts do not exist. As an example, a URI-identified resource may be used as an object or subject in one context, and it may be used as a predicate in another context. According to RDF this is correct.

In essence, RDF provides a formal semantics which is not aware of the concepts of entities and relations and therefore cannot differentiate between them. This is quite different to other ontology modelling methods, such as Description Logics in general, or OWL as one of their concrete implementations. Using RDF, facts can be easily captured, but not structured in form of an ontology.

3.2.1.1.1 Modelling Ontologies

In order to properly model ontologies syntactically and semantically in RDF, the Resource Description Framework Schema (RDFS) specification has been released. For ontology modelling, RDFS defines a controlled vocabulary of subject/object and predicate resources whose semantics is formally defined. The set of these subjects/objects and predicates provides comparably low expressiveness, such that only light-weight types of ontologies can be modelled which in consequence only offer restricted ontology reasoning capabilities. In the following some of the defined predicates in RDFS are introduced.

- "rdf:type": This predicate-type resource is used to explicitly declare the type of a given resource (i.e. the class this resource belongs to). For example, the triple "(ProcessNumber1 rdf:type Process)" asserts that the resource "ProcessNumber1" is of type "Process" (which informally is the class of all processes). This predicate is also used to construct new types/classes. In our example, in case the

[2] http://www.w3.org/XML/Schema

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

class "Process" has not been defined before, the type would now be part of the RDF knowledge base, in which the class contains one individual, the process "ProcessNumber1".

- "rdfs:class": This subject/object-type resource is the set of all classes defined in an RDF knowledge base. In combination with the "rdf:type" predicate, it can be used to explicitly declare a new class. For example, the triple "(Process rdf:type rdfs:class)" asserts that the resource "Process" is a new type (which informally is the class of all processes).
- "rdfs:property": This subject/object-type resource is the set of all property types (relations in general ontology terminology) defined in an RDF knowledge base. In combination with the "rdf:type" predicate, it can be used to explicitly declare a new property type. For example, the triple "(Process rdfs:type rdfs:property)" asserts that the resource "Process" is a new property type (which informally is the class of all processes).
- "rdfs:subClassOf": This predicate-type resource is used to construct class hierarchies by explicit declaration. In other words, it is used to explicitly declare sub-classes of already declared, or defined, classes. For example, the triple "(EuProject rdfs: subClassOf Project)" asserts, i.e. declares, that the resource "EuProject" is a sub-class, or sub-type, of the "Project" resource (which informally is the class of all projects). This sub-class relation means that every individual which is an instance of the class "EuProject" also is an instance of the class "Project". But not vice-versa, hence the sub-class relation.
- "rdfs:subPropertyOf": analogous to classes, the "rdfs:subPropertyOf" predicate-type resource can be used to establish hierarchies of properties. For example, the triple "(isCeoOf rdfs:subPropertyOf worksFor)" expresses that everything that leads an entity also works for this entity.
- "rdfs:domain" and "rdfs:range": In the last expression it would have been nice if we could express that only persons can lead something and that only persons can work for something. In addition, it would have been nice to express that this something is actually a company. The predicate-type resources "rdfs:domain" and "rdfs:range" can be used to constrain the domain and range of a property and all its sub-properties, as (obviously) they inherit these constraints. For example, the triple "(worksFor rdfs:domain Person)" constrains the "worksFor" property's domain to persons (if and only if the class "Person" contains all and nothing-but persons modelled in our ontology).

Using these resources to declare hierarchies of classes, hierarchies of properties, and characteristics of properties (domains and ranges) is sufficient to capture the ontologic modelling capabilities of RDFS. Additional semantics, which might seem natural to us, cannot be modelled using pure RDFS. For example, we cannot express that a company needs to have a leader, or that two classes are disjoint from each other, e.g. that a person cannot be a company and vice-versa.

3.2.1.1.2 Reasoning on Ontologies

Nevertheless, the formal semantics of RDFS provide correct and complete reasoning capabilities that are helpful even though RDFS is restricted in its expressiveness. For example, ontology entailment can be used to test if an ontology is entailed by another ontology, for example, to test whether a statement is satisfied by an ontology. For example, you might want to ask your knowledge base whether scientists are intelligent. The ontology entailment can test whether this (mini-)ontology is entailed by your knowledge base, i.e. test whether the statement is generally true.

3.2.1.2 OWL

The OWL standards (v1.0 and v2.0) are two consecutive approaches from the Semantic Web community which are based on a small subset of the family of Description Logics (DL). Each DL is a restricted part of predicate logic, whereby all description logics have in common that they are decidable (which predicate logic is not) to make them practically tractable.

The OWL v1.0 standard comprises three dialects, called “full”, “DL”, and “lite”. In practice, the “full” and “lite” standards are of minor importance. OWL-Full has the particular disadvantage of being undecidable, due to the highly expressive features that have been poured into its backing description logic. To the best of our knowledge, there does not exist a single implementation of OWL-Full (which could be used to manage OWL-Full ontologies and reason on them). From a practical point of view, OWL-Full is therefore irrelevant. In contrast to OWL-Full, OWL-Lite has been designed for users and application with only few requirements on expressiveness. It is a comparably compact language, which provides comparably efficient reasoning capabilities. Nevertheless, after OWL v1.0 had been released, it was discovered that it is only marginally more efficient than OWL-DL but is remarkably less expressive.

In consequence, the OWL-DL standard is the one paid most attention in academia and industry. Therefore, the remainder of this chapter will focus on OWL-DL (with its latest release in OWL v2.0). OWL-DL has the following general characteristics:

- Expressiveness: Various complex situations can be modelled in OWL-DL.
- Decidability: The reasoning mechanisms of OWL-DL provide correct results in tractable time, depending on the size of the input.
- Practicability: There are mature implementations for OWL-DL ontology management and reasoning available on the commercial market and as free software, which provide practically acceptable response times to relevant real world problems.
- Declarative Semantics: The meaning of an expression does only depend on the expression itself, and not on the semantics implicitly given by an implementation of the ontology management or reasoning systems. In addition, it is not necessary to understand the inner workings of the reasoning mechanisms.
- Standardization: Based on the international standardization of OWL by the W3C consortium, the syntax and semantics of OWL are globally defined. This enables and fosters collaboration and integration of ontologies, among other activities.

In summary, OWL is founded on a family of logics called Description Logics (DLs). The family of Description Logics is a successful set of mathematical logic-based ontologic knowledge representation languages, which unify the distinct characteristics of a formal syntax and a model-theoretic formal semantics which enables sound and complete reasoning on ontologies. Additionally, the reasoning capabilities are decidable and therefore practically tractable.

To provide an example of a DL-ontology employed in practice: the SNOMED (Systematized Nomenclature of Human and Veterinary Medicine) [RED SNOMED] is a complex ontology which contains more than 800k medical terms which define 300k concepts and their relations. The SNOMED ontology is used throughout the US health care system for diagnosis and analysis purposes.

Each description logic is characterized by its set of constructors. As introduced above, OWL represents the current state-of-the-art after a long period of development, and its OWL-DL language contains many constructors to foster its quite high expressiveness. In the following, we illustrate the expressiveness of OWL-DL by iteratively introducing subsets of the description logic that backs OWL-DL.

3.2.1.2.1 Modelling Ontologies

There are various syntactic forms for representing ontologies in OWL. The “traditional” syntax in description logics employs symbolic notation known from mathematical logics, e.g. “¬Female”. In comparison, the syntax standardised by the W3C consortium is a prefix notation grounded in set theory, e.g. “IntersectionOf(Rich, Famous)” or “AllValuesFrom(has-child Female)”. As both of these notations have shown to have a steep learning curve for beginners, this document uses the Manchester OWL syntax, which is an infix notation.

The most fundamental description logic subsumed by OWL-DL is ALC which was already designed in the 80ies. ALC provides the traditional Boolean logic connectives (“and”, “or”, “not”) and universal and existential quantification for describing the individuals of a class.

Table 2: Constructors in ALC

Expression	Semantics
Rich “and” Famous	The set of all individuals which and rich and famous.
Poor “or” sick	The set of all individuals which are either poor or sick (or both).
“not” Female	The set of all individuals which are not female
has-child “only”	The set of all individuals which have only one child (or no children at all)
has-child “some” Doctor	The set of all individuals which have at least one child that is a doctor (they may have more children)

In the 90ies, the description logic SHIQ has been designed to extend the expressiveness of ALC by four features:

- “Number restrictions” provide the ability to restrict the number of associated individuals in a role.
- “Inverse roles” provide role statements that are inversed, i.e. they restrict the “left side” of a role, instead of the “right side” as was only possible with ALC.

- “Role inclusions” provide the ability to construct hierarchies of roles, which additionally can be declared as being “transitive”.

Table 3: Additional Constructors in SHIQ

Expression	Semantics
has-child “min 5” Male	The set of all individuals that have at least five male children.
“inverse” has-child “some” Rich	The set of all individuals which have at least one rich parent.
has-successor “Characteristics: Transitive”	For example, the successor A of individual B is also the successor of C, if C is the successor of B.
has-child “SubPropertyOf: has-successor”	For example, all of my children are also my successors (in case successor is interpreted as “descendant”).

In the meantime, between the design of the SHIQ description logic and the foundation of the Semantic Web on description logics, many useful constructors and algorithms have been designed, which were both integrateable into the Semantic Web stack. Therefore, the OWL-DL v1.0 release also contains nominals, i.e. classes which just contain a single individual, and data types (conceptually classes of a finite or infinite number of individuals) to associate concrete values with individuals (e.g. the age of a person). Furthermore, OWL-DL v1.0 also provides the ability to declare roles as being symmetric.

It is possible to associate numbers with individuals in an ontology without facilitating data types, for example, by incorporating the individuals “five” and “eight” into the ontology (which represents the integer number 5 and 8) and associating them to other individuals, e.g. using the age relation. But, in this way, only a finite number of integer numbers can be represented, and it is hard to infer that “eight” is “larger than” “five” (in case this has not been explicitly defined for every pair of integer numbers in the ontology). In comparison to this manual notion, a data type consists of a set of defined individuals (e.g. the set of integer numbers) with which a relation is defined (e.g. a total order in case of the integer numbers). In OWL, individuals in an ontology can be related with values of a data type using so-called data properties (as opposed to object properties), e.g. using a “has-age” property, a person could be related to an integer number that reflects her/his age.

Table 4: Additional Constructors in OWL-DL v1.0

Expression	Semantics
UsPresident “EquivalentTo: { Obama }”	The class “UsPresident” consists of only one individual, “Obama”.
has-length “exactly 5”	The set of all individuals which have a length of exactly 5.
has-price “max 20”	The set of all individuals having a price of max. 20
has-age “min 50”	The set of individuals which are at least 50

married-to "Characteristics: Symmetric"	For example, if Hans is married to Grete, then Grete is also married to Hans.

After the OWL v1.0 standard had been released and had been implemented in practical systems, new end-user-driven requirements on the expressiveness of OWL ontologies came up, which eventually led to the design and release of the OWL v2.0 standard. For example, OWL v2.0 provides the ability to chain a restricted form of rule, called property, and to declare additional characteristics of properties: transitivity, symmetry and reflexivity. Furthermore, it is also possible to declare two properties to be disjoint, analogous to roles.

Table 5: Additional Constructors in OWL-DL v2.0

Expression	Semantics
owns "SubPropertyChain: owns °" has-part	For example, if I own something, than I own also all parts of it
has-child "DisjointWith: married-to"	For example, somebody cannot be married to his/her children
has-child "Characteristics: Asymmetric"	For example, my child cannot be my farther
knows "Characteristics: Reflexive"	Each individual knows it-/his-/herself
has-child "Characteristics: Irreflexive"	It is impossible that somebody or something is the child of it-/his-/herself

An OWL v2.0 ontology basically consists of sets of the following types of axiomatic declarations:

- "Class declarations": For each class declaration, it is possible to declare sub-classes, equivalent classes and disjoint classes.
- "Object properties declarations": For each object property, sub-properties, property chains, equivalent and disjoint object properties can be declared. Besides this, the domain and range of an object property and its inverse property can be restricted. Furthermore, characteristics of a property, such as symmetry, transitivity and reflexivity can be declared.
- "Data property declarations": For each data property, sub-properties, equivalent and disjoint data properties can be declared. Besides this, the domain and range of an data property and its inverse property can be restricted.
- "Individual declarations": For each individual the classes it belongs to, relations to other individuals or data can be declared. In addition, equivalence or in-equivalence between individuals can be declared.

Some fundamental design decisions of OWL resulted in restrictions of its expressiveness. The fundamental requirement that all reasoning problems in OWL have to be decidable (i.e. any combination of OWL operators still represents a decidable problem), restricts expression abilities. Furthermore, the goal of OWL is to only contain practically relevant and algorithmically efficient operators. In consequence, for example, OWL does not contain Boolean operators. But there are extensions of OWL available: for example, the

Semantic Web Rule Language (SWRL) which provides the ability to formulate various rules on individuals, classes and properties. In addition, as stated earlier, OWL is basically grounded in predicate logic. Therefore it is only possible to declare “certain knowledge”. It is not possible to declare “uncertain knowledge” (e.g. default logic), neither probabilistic nor possibilistic (e.g. fuzzy logic). This could for example provide the ability to reason about the “similarity” of individuals or classes, which is a feature not provided in OWL.

3.2.1.2.2 Profiles

On the one hand, OWL is a quite expressive language which allows users (i.e. domain modellers) to model complex situations in their respective domain of interest. On the other hand, this expressiveness is not required in every use case and may even lead to problems. Firstly, users with little experience in ontology modelling and formal logic may have issues in properly using the constructors which lead to modelling inaccuracies and problems. Secondly, the reasoning mechanisms for this expressiveness may have in some cases quite high processing time and memory complexity, e.g. solving the satisfiability problem for classes is NexpTime-complete.

As mentioned above, the earlier-mentioned OWL-Lite standard has not been accepted in practice, even though it provides higher performance than OWL-DL at the cost of less expressiveness, as the performance gain is not big enough to justify the expressive loss. Therefore, in OWL-DL of OWL v2.0, a more flexible approach has been introduced: fragments of OWL-DL, called „profiles“ have been defined, which provide less expressiveness than OWL-DL, but more efficient reasoning mechanisms. For example, OWL 2 EL and OWL 2 QL are prominent examples of such profiles.

3.2.1.2.3 Reasoning on Ontologies

Implicitly given knowledge refers to knowledge which can be inferred from explicitly given knowledge using generic inference mechanisms. This is a particular characteristic of ontologies that is not provided by databases and is the fundamental difference between an ontology and a database.

One could imagine designing implementation-specific applications that provide the ability to infer knowledge from the explicitly given knowledge in a database. But these applications do not provide the ability to infer knowledge in general. In particular, knowledge whose way of inference is not clear at application design time cannot be inferred by these applications as it requires a generic inference mechanism. In addition, such applications are generally bound to their specific databases. As a short remark: an ontology could be implemented as a database, where the reasoning mechanism implementations operate on the database.

For the purpose of generic knowledge inference on ontologies, various generic reasoning (or inference) mechanisms for ontologies have been designed. The most prominent ones are illustrated in the following. But for understanding the following reasoning techniques, the notion of a “model” of an ontology is important. A “model” consists of a set of individuals which are associated with the classes in the ontology, such that all axioms (e.g. constraints) in the ontology are satisfied. In other words, a model is a set of individuals which satisfies the ontology, or, a model describes a valid instance of an ontology.

- “Ontology consistency”: verifies whether there exists at least one potential model which satisfies all axioms in an ontology. This reasoning mechanism is useful for detecting modelling mistakes during ontology knowledge engineering.
- “Ontology entailment”: checks whether all potential models for an ontology A are also valid models for another ontology B. This would mean that ontology B entails ontology A. In other words, the ontology B implies ontology A, or, the knowledge stated in ontology A is supported by the knowledge in ontology B. In case all models of ontology B also satisfy ontology A, the two ontologies are “equivalent”. This reasoning mechanism can for example be used to verify that a change to the ontology (e.g. for efficiency reasons) does not modify the semantics of the entire ontology.
- “Satisfiability of classes”: verifies whether there exists at least one potential model which satisfies all axioms in an ontology and in which all classes of the ontology contain at least one individual. This reasoning mechanism is also useful for detecting modelling mistakes during ontology knowledge engineering, as knowledge engineers do not want to model classes of individuals which actually cannot contain a single individual.
- “Class subsumption”: checks whether all individuals of a class A also have to be individuals of another class B. This reasoning mechanism can be used to help create class hierarchies in an ontology, for example, by identifying equivalent classes, that are duplicated in your current hierarchy. In this sense it also assists in detecting modelling mistakes.
- “Instance test”: checks whether an individual (or instance) has to be an instance of a class. This reasoning mechanism can be used to infer knowledge about individuals. For example, if it is determined that an instance is an individual of a class, which was not known before, then the individual inherits all properties (or relations) of that class. If we would determine that “ProcessNumber1”, besides being a process, is also a business process, we could infer that the process has to run on a business process engine (given the background knowledge that all business processes by definition have to run on a business process engine).
- “Queries”: queries are formulae consisting of of classes, properties, free variables and logical operators (e.g. the traditional logical operators: “and”, “not”, “or”), for which the reasoning mechanism tries to find individuals in the ontology that match the free variables and satisfy the logical formulae, so that the logical formula evaluates to “true”. For example, the query “(Student(x) AND enrolled-in(x,y) AND teaches(MarvinMinsky,y))” can be satisfied by an ontology which sets the free variables “x” and “y” for example to “x:=JaneDoe” and “y:=ArtificialIntelligence”.

This list of reasoning mechanisms on ontologies describes all mechanisms covered by the OWL standard. Besides these, there are many non-standard reasoning mechanisms in the literature and probably many to come in future, for example in TIMBUS.

3.2.2 System and Software Dependencies

Capturing system and software dependencies is an important aspect in capturing the relevant context parameters of a business process. Therefore, this section presents the Common Upgradeability Description

Format (CUDF (Treinen and Zacchiroli, 2008)) and the Virtual Resource Description Framework (VRDF, 2011) standards. CUDF is and can be used in Linux distributions for describing dependencies at the software package level of abstraction. VRDF is captured only briefly, though, as its details are not yet officially released, however it addresses the issue of dependencies in large-scale networks with virtualised resources which is in the scope of the work in Tasks 4.5 and 4.2.

CUDF is a generalised approach for modelling inter-dependent software packages of GNU/Linux based computing systems and the approach provides a set of specialised (semi-)formal semantics to answer the upgrade problem. The upgrade problem (or upgrade request when being performed) is where a user or computer attempts to install, remove, downgrade or upgrade packages to a GNU/Linux Operating System.

3.2.2.1 CUDF

CUDF (Common Upgradeability Description Format) and DUDF (Distribution Upgradeability Description Format) are formats for describing upgrade scenarios in package-based Free and Open Source Software (FOSS) distributions. More information about CUDF can be found at http://mancoosi.org/cudf/primer/.

CUDF was designed to capture and express upgrade problems in a format that is independent of the type of GNU/Linux Operating System and allows for a class of software tools, known as solvers, to work on identifying possible solutions for upgrading a set of packages requested by the user. The solvers attempt to identify whether a user-selected set of packages can be upgraded based on what is available remotely (the repository), locally installed and the dependency relations encoded in Linux packages. Different solvers work in different ways for reaching a solution and there have been many International Competitions to rank and compare their performance (http://mancoosi.org/misc/), but at the basis they all use the same formalism and upgrade problems as their inputs. The DUDF format is generated on a per distribution basis. Each distribution that intends to produce DUDF compliant files has to modify the package installer or provide a set of scripts for capturing the upgrade problem. This is dependent on the type of installer that is being used on the distribution. The tools identify the corpus of packages that are available to the installer at the time the upgrade request is made. The full requests are then captured in a standardised manner and submitted to a centralised server based on the distribution. The distribution servers collate the information and convert the DUDF files into CUDF. The CUDF files are more abstract and describe relationships between packages at a higher level than DUDF. The CUDF representation is then submitted to a centralised repository where the upgrade problem sets can be collected and used as inputs for the solvers.

The important part of CUDF for context analysis is that the tools capture the state of the system in terms of software packages in a standard format. For TIMBUS, the upgrade request, when capturing the state of the system, is less important but when the reasoning system will be used to infer whether certain dependencies are met, the solvers can use upgrade requests and the existing tools to examine if the software on the system can be upgraded based on the constraints and criteria provided. It also provides an upgrade plan that can be used if the packages on the system should be upgraded to meet the specified criteria.

DUDF and CUDF by design have been left extensible. It is proposed that, within the TIMBUS project in Task 4.2 and D4.2/D4.3, this format will be the basis for describing software dependencies and, as such, will be a format that the context analysis tools will need to be able to interface with.

3.2.2.2 VRDF

The Virtual Resource Description Framework[4] is a framework developed to describe and analyse complex dependencies in the context of cloud computing. As such, it aims at representing dependencies of services and virtualised infrastructure (such as virtual machines or virtual networks) to the physical infrastructure. To this end, it provides an RDF schema to model connections, devices, networks and other related entities. This model shall then allow to, for example, assess the impact of failures of physical hardware on the virtual infrastructure.

3.2.3 Processes

Capturing process models (e.g. causality and timing) has also been identified as an important aspect in capturing the relevant context parameters of a business process. Therefore, this section presents several process modelling techniques.

3.2.3.1 Business Process Model and Notation

The Business Process Model and Notation (BPMN) is a graphical modelling language developed by the Object Management Group (OMG) for the specific representation of business processes (Object Management Group, 2011). BPMN models can be used for the purpose of documentation or for execution by a BPM platform. Depending on the purpose, models can be more or less detailed.

The elements that can be modelled in BPMN include: (*i*) Flow Objects, which define the behaviour of a business process (i.e., Events, Activities, and Gateways); (*ii*) Data (e.g., Data Objects, Data Inputs, Data Outputs, and Data Stores); (*iii*) Connecting Objects, which connect Flow Objects to each other or to other elements (e.g., Sequence Flows, Message Flows, Associations, and Data Associations); (*iv*) Swimlanes, which group modelling elements (e.g., Pools and Lanes); and (*v*) Artefacts, which provide additional information on a process (e.g., Group and Text Annotations).

[4] http://sw-app.org/pub/sfsw07-vrdfgraph.pdf

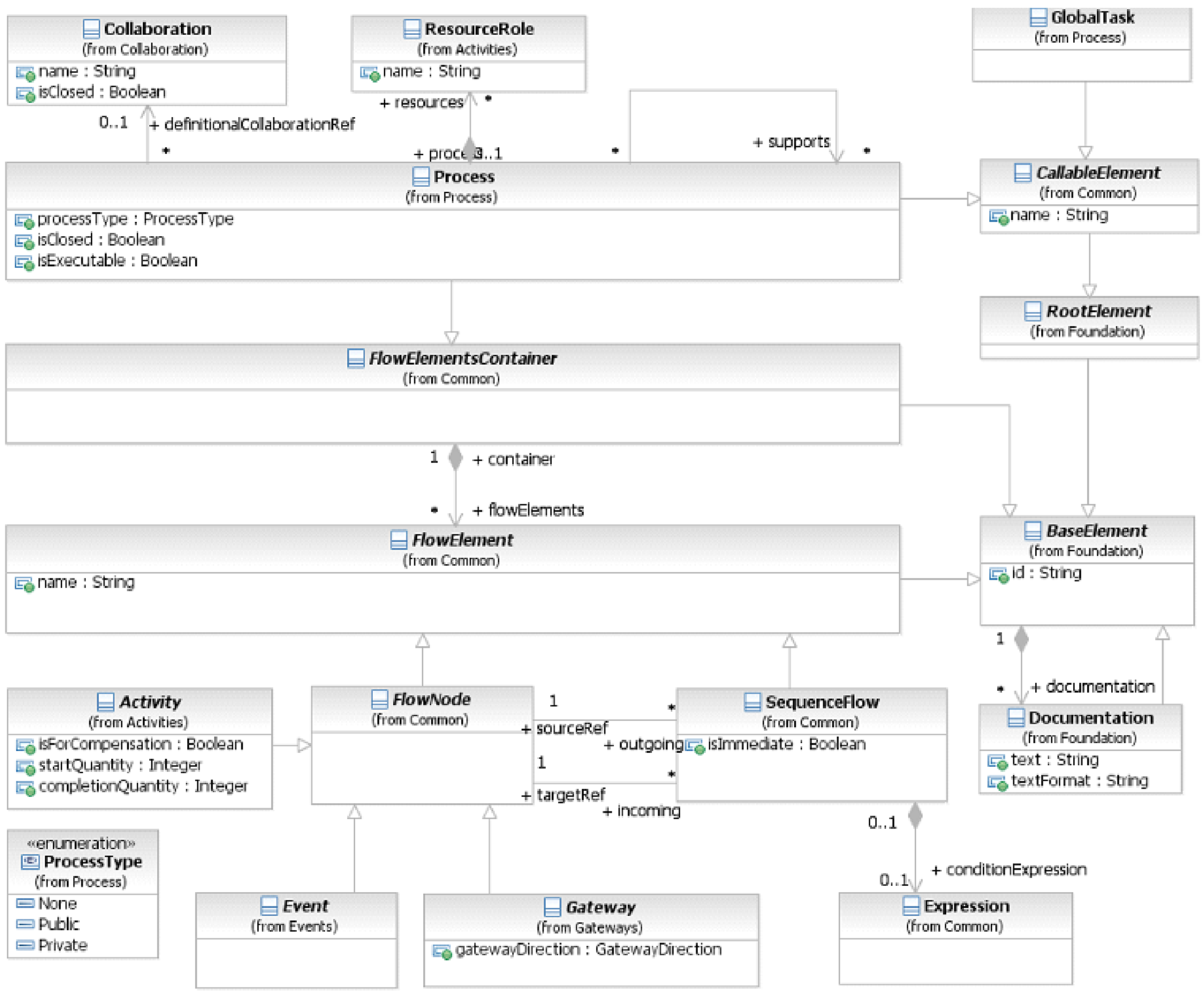

Collaboration (from Collaboration)
name : String
isClosed : Boolean
ResourceRole (from Activities)
name : String
GlobalTask (from Process)
+ resources
0..1
+ definitionalCollaborationRef
+ supports
Process (from Process)
processType : ProcessType
isClosed : Boolean
isExecutable : Boolean
CallableElement (from Common)
name : String
RootElement (from Foundation)
FlowElementsContainer (from Common)
+ container
+ flowElements
FlowElement (from Common)
name : String
BaseElement (from Foundation)
id : String
+ documentation
Documentation (from Foundation)
text : String
textFormat : String
Activity (from Activities)
isForCompensation : Boolean
startQuantity : Integer
completionQuantity : Integer
FlowNode (from Common)
+ sourceRef
+ outgoing
+ targetRef
+ incoming
SequenceFlow (from Common)
isImmediate : Boolean
+ conditionExpression
«enumeration» ProcessType (from Process)
None
Public
Private
Event (from Events)
Gateway (from Gateways)
gatewayDirection : GatewayDirection
Expression (from Common)

Figure 10: BPMN Meta-model Fragment

Source: BPMN Specification

BPMN models allow the observation of the direct context of a business process. Figure 10 depicts a fraction of the BPMN meta-model dealing with the definition of process. It can be observed from the figure that a process might be comprised of several flow elements, such as Activities, Events, and Gateways, which form a sequence flow. Resources can be assigned to processes. A process might take part in a collaboration and should also comprise documentation.

3.2.3.2 Event-Driven Process Chain (EPC)

Event-driven process chains are a way to model business processes, which is used in the enterprise modelling approach ARIS. A process is considered as a sequence of events and functions. Events describe the state of the process (orange hexagons in Figure 11), whilst functions (green rectangles in Figure 11) lead to a new state. There can also be organisational units assigned to the functions, as well as information objects like databases or systems (blue rectangles in Figure 11 depict the involvement of an SAP system). Connectors controlling the flow of the process (AND, OR, XOR) similar to BPMN are available.

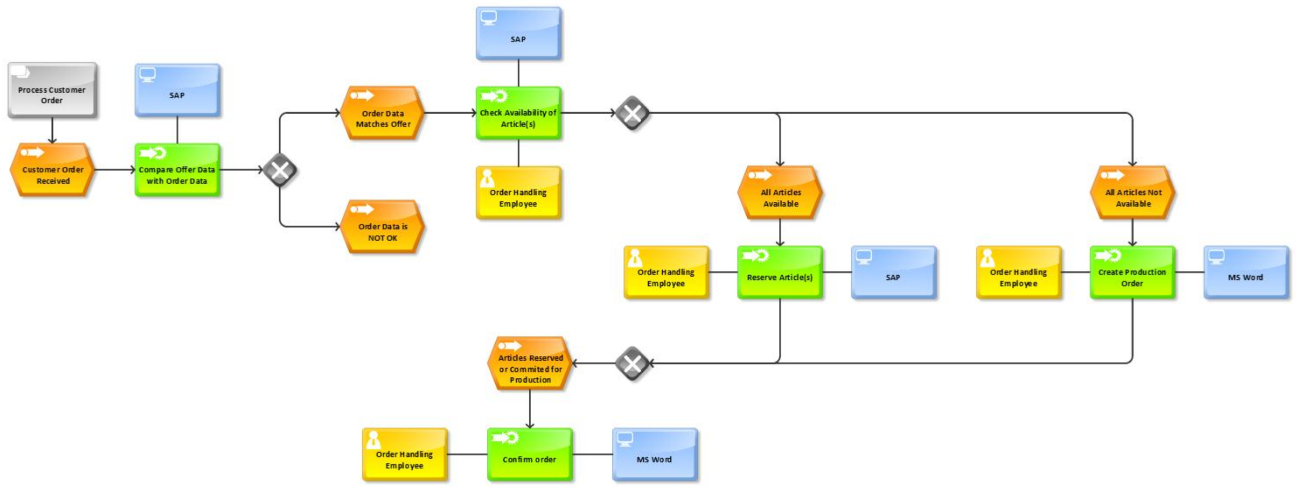


Figure 11: EPC model

Source: http://www.ariscommunity.com/users/insight/2011-12-15-sap-model

3.2.3.3 WS-BPEL (Business Process Execution Language)

WS-BPEL is an XML-based language for orchestrating web services, i.e. combining them in a way so that they form an executable business process. While there are also proposals for a graphical notation as in BPMN the main focus of BPEL lies in describing how the business process is executed.

The resulting business process consists of activities which are represented by web services. These services can either be running in the same company in which the resulting business process is executed or provided by external service providers.

The resulting business process consists of activities which are represented by web services. The process itself can also be seen as a web service.

3.2.3.4 Petri-Nets

Unlike the other discussed modelling languages, Petri nets were not specifically introduced to describe business processes, but for chemical processes. In addition to business process modelling they are widely used in different areas like mechanical engineering or logistics.

The theory of Petri nets has been researched extensively in mathematics. Petri nets consist of very few different types of model elements: places and transitions that are connected by arcs and a set of tokens that are distributed over the places. This distribution describes the state of the net. As Petri nets come with a simple execution semantic they are easy to use for business process simulation, bottle neck identification and optimization (van der Aalst, 2002). In its common semantic a transition is enabled if every one of its input places (places connected to it via ingoing arcs) contains at least one token. A transition then "fires", i.e. decrements the number of tokens in every one of its input places and increments it in every one of its output places. Mapping business process semantics to Petri nets, transitions stand for process steps and the way they are connected to each other via arcs and places is describing the control flow. The tokens can be considered as process instances (e.g. a customer request that is being handled in a CRM process).

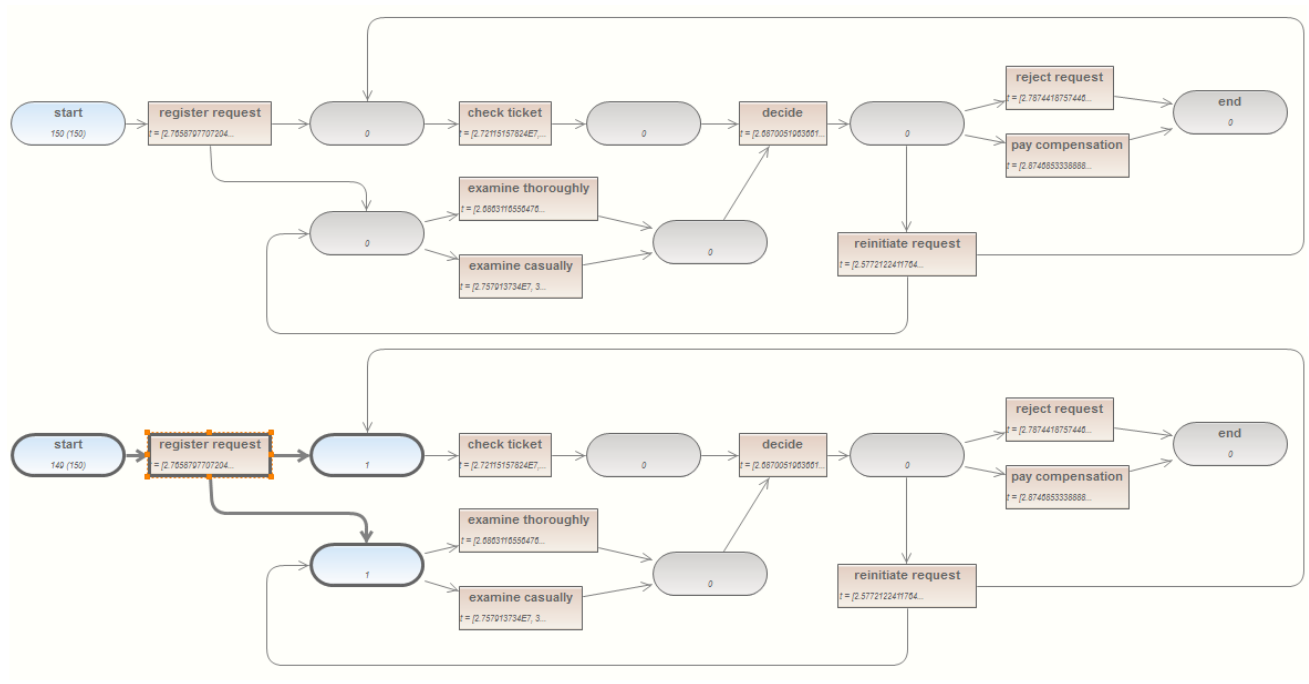


Figure 12: Petri net before and after a state change

Figure 12 shows a process in a customer relationship management system of an airline where a customer requests compensation for a cancelled flight. The process consists of 8 steps from registration to decision and finally payment/rejection of the request. To illustrate Petri net semantics the same net is shown in 2 states, an initial state (upper net) and in a state that is taken after the execution of the process step "register request" (lower net). Initially there are 150 tokens in the "start" place and 0 in all of the other places making only the transition "register request" enabled. After execution of "register request" there are 149 tokens left in the start place and 1 in both of the output places of the transition. That also shows an example of a parallel execution semantic in business processes, as there are now 3 transitions enabled making it possible to execute "check ticket" and either "examine casually" or "examine thoroughly".

3.2.4 Digital Preservation

This section of the deliverable will describe some of the existing context modelling standards and techniques which are either available today or being developed as part of other research projects. This section also calls out where TIMBUS has active contacts with these bodies but that work is detailed more fully in the exploitation report for year 1.

Metadata is any type of data which is used to describe other data. An index of a document could be considered to be metadata because it is information which describes or relates to the data in the main body of a document. There are of course many types of metadata, but the TIMBUS project is concerned with a special branch called digital preservation metadata. This is metadata which is essential to ensure the long-term accessibility and intelligibility of digital resources. The expediency tools to be researched in TIMBUS are all generators of metadata. Metadata therefore places digital objects and resources in a context which makes them more easily understandable. TIMBUS concentrates its efforts on this set of this context metadata with a special focus on identifying and creating parameters which will improve the industry's long term data retention capability in the area of business process preservation. To do this, TIMBUS has carried out a study of relevant bodies, industry standards and other research projects and has either formal or informal ongoing contact with all these as detailed in the exploitation and dissemination reports.

Discovering what DP metadata to store is a difficult challenge. Due to the relatively short life of the digital age, these issues are only beginning to emerge which means the IT sector has very little experience in validating the longevity of digital objects. It is difficult because of the uncertainty of the future, specifically in the areas of technology advancement and legal/regulatory changes. The challenge arises because the use of technology itself introduces a barrier which inhibits the human beings natural ability to access or understand information without the availability of compatible technology platforms and the knowledge (or context) of how to operate and access that information.

At the POCOS conference in 2012, Angela Dappert gave a presentation on how TIMBUS is investigating how to do this for more complex data objects than previously attempted. The remainder of this section of the deliverable references that presentation. In the case of TIMBUS, to preserve a business processes, we need to preserve not just the data that resided in the business process but also the software that was required to render, process or otherwise view it in the future in a way which insulates software from underlying hardware changes. While there are many software repositories for current generation applications (such as SourceForge, or any Linux repository) there are very few software repositories in the world which attempt to do what TIMBUS does and gather the types of meta data which is needed for this. The repositories which do exist (for example emulation software for Amega or Commodore games as on http://AMINET.net) have been preserved by amateur gaming enthusiasts or professional ones have limited metadata models and are too purpose specific to be generally applied. For example, the National Software Reference Library (NIST, 2012) in the US who collect software and file profiles into a reference data set of information which can be used by law enforcement agencies to determine which files on computer systems are important as evidence in crimes.

TIMBUS also needs to capture metadata. The Software Sustainability Institute[5] produced a report (Matthews, McIlwrath, Giaretta, and Conwa, 2008) that identified seven significant property categories of software metadata which should be captured in order to preserve that software. These seven were functionality, SW composition, provenance & ownership, user interaction, SW environment, SW architecture and operating performance. For each category, four property classes exist, package, version, variant and download. Table 6 below shows an example of what these might look like for the software interaction category.

[5] http://software.ac.uk/

Table 6: Examples of SW Interaction Metadata

Source: POCOS conference 2012, A.Dappert, http://vimeo.com/36101909

Category	Package	Version	Variant	Download
Software Interaction	SW Overview Tutorials Requirements	Source Manual Installation Test Cases Specification Functional evolution	Binary Source Configuration Operating system Programming language	File

To understand the challenge of what might be required to preserve a business process more fully, this deliverable will now present a digital asset migration example. Taking the TIMBUS approach, it will show the reader how difficult a problem this can be and what types of metadata need to be captured.

TIMBUS performs analysis in four categories which ultimately result in the gathering of context data categories, namely these are ERM (what was preserved and why based on understand of risks), Dependency Analysis (understand what components have dependencies which may need to be allowed for), Business Process Context Capture (how does the business process fit together) and Legalities Lifecycle Management (regulatory, intellectual property aspects relating to preserved data). The example is based on the migration of TIFF image files to a smaller file format within an existing archive maintained by a memory institution. It has 10 phases as shown, ingest of TIFF files, quality analysis to ensure they are not faulty, decide the migration format (JPG, etc), choose migration tool, perform the actual migration, then run automatic quality analysis, followed by manual quality analysis. At that point the data is stored, the original data is removed and access to the data is ensured. Some ERM metadata examples which could occur at each of these steps are shown. Figure 13 below shows ERM risks or potential errors that could happen at each step which might result in problems later on or possibly even in the corruption or loss of data. These can help identify which steps need to be preserved and why.

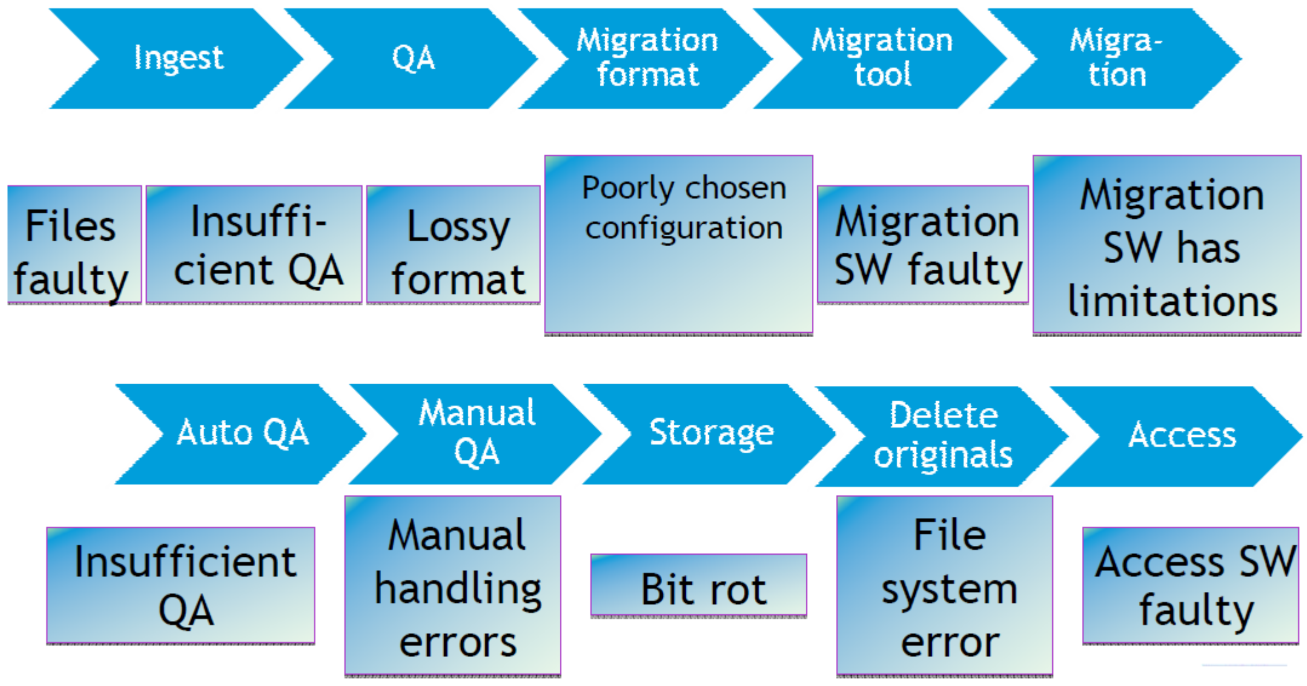


Figure 13: ERM Context Capture

Source: POCOS conference 2012, A.Dappert, http://vimeo.com/36101909

Next it is necessary to understand the dependencies between various steps in the workflow. This will help identify which parts of the software environment need to be preserved by building a dependency map.

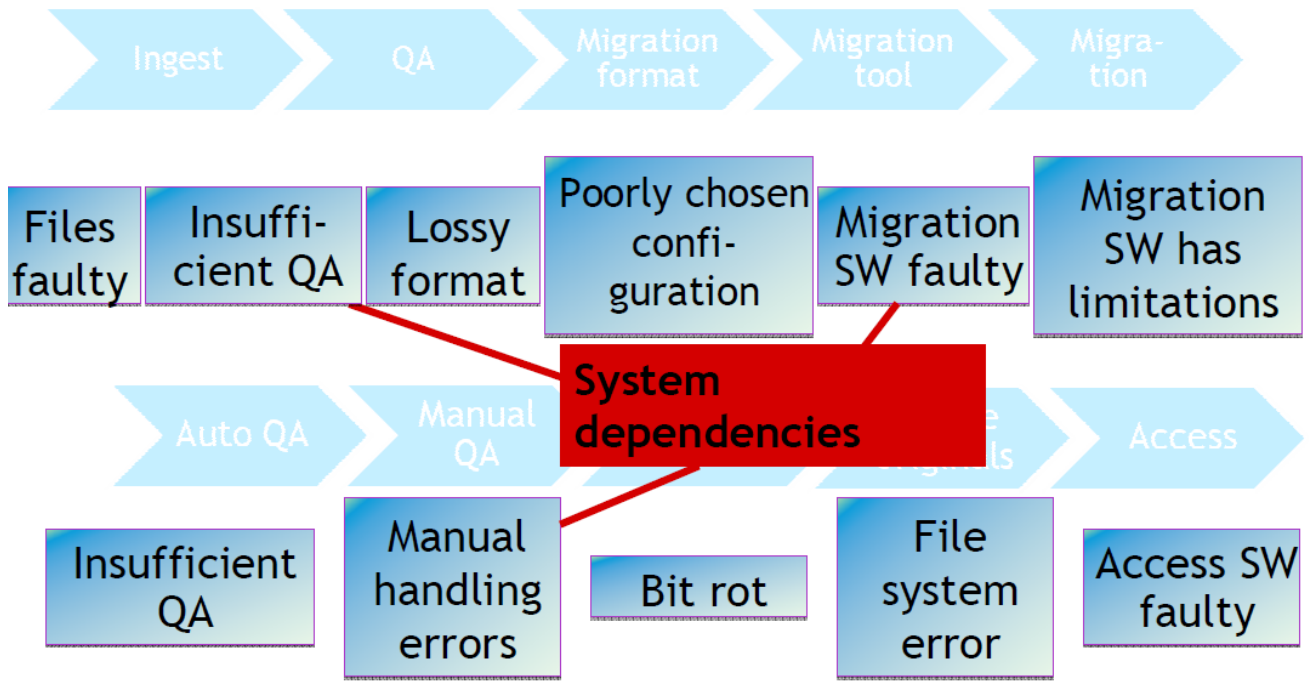


Figure 14: Dependency Context Capture

Source: POCOS conference 2012, A.Dappert, http://vimeo.com/36101909

BP context capture must determine what information is needed later on to understand causes of possible corruption or loss of data. In this example, it may be necessary to record information at the QA step such as sampling method and tests used, requirements and policies, logs and other specifications.

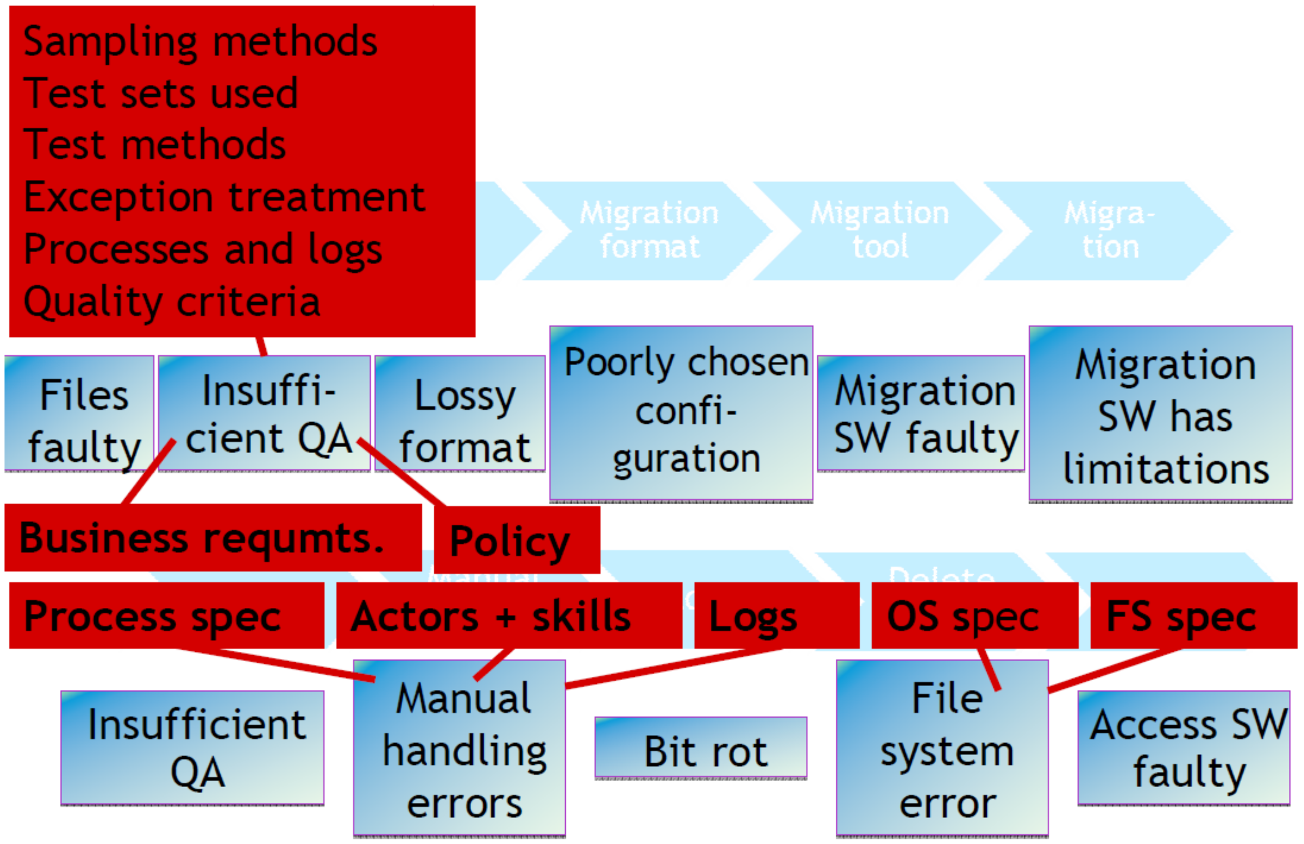


Figure 15: BP Metadata Capture

Source: POCOS conference 2012, A.Dappert, http://vimeo.com/36101909

The last part of the metadata capture shows some legality examples of what might be required for the preservation of a business process such as intellectual property restrictions, software licensing and ownership, and regulatory data protection.

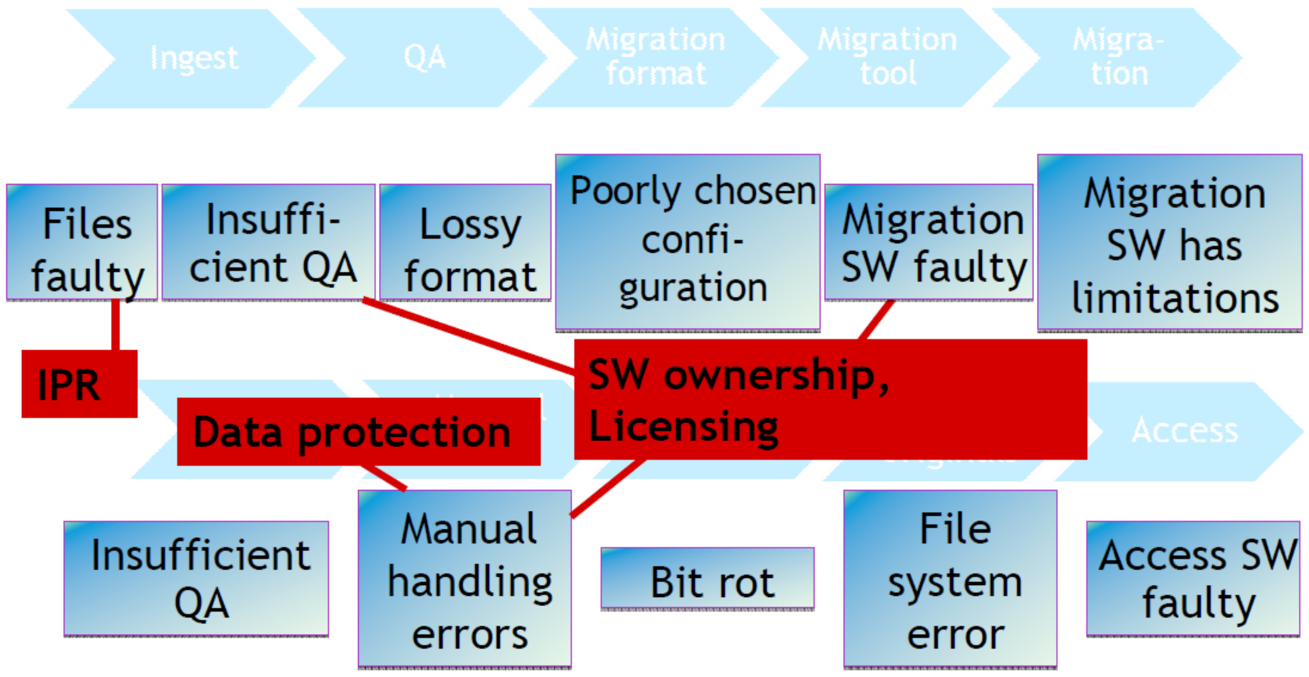


Figure 16: Legalities Metadata Capture

Source: POCOS conference 2012, A.Dappert, http://vimeo.com/36101909

3.2.4.1 PREMIS Data Dictionary

PREMIS (The Library of Congress, 2012) is the first standard that this section of the deliverable will discuss. PREMIS is an acronym for PREservation Metadata Implementation Strategies. It is an implemented standard for a metadata model that specifically seeks to identify and classify all possible metadata types (or contexts) that are required for long term accessibility and intelligibility of digital objects. The standard was first created in 2005 when a working group was created by RLG (Research Libraries Group) and OCLC (Online Computer Library Center) (OCLC President, 2006). RLG have since merged with the OCLE who are a non-for profit research organisation whose computer systems help libraries and other research institutions around the world. The PREMIS standard working group continue to publish advances in their work through the the Library of Congress website[6] in the United States. The Library of Congress is a classic memory institution and its members were heavily involved in the first draft of PREMIS (PREMIS, 2005). It is the oldest federal institution in the US with a mission of carrying out and supporting research, promoting appreciation of literature. It is of course concerned with acquiring, cataloguing and preserving various library collections. In this role, they maintain, contribute or consume 23 standards in their current operations, of which PREMIS is one.

The PREMIS data model entities and the relationships between them are shown below in Figure 17. Each metadata type which is defined in PREMIS is a property of one of these. Environment is illustrated as a property of objects. It is a living model which continues to be evolved as required and there is currently an

[6] http://www.loc.gov/about/

PREMIS work group developing the Environment property with assistance from the TIMBUS project. The environment entity will describe what is needed to render or use an object in terms of operating system, application software or compute resources.

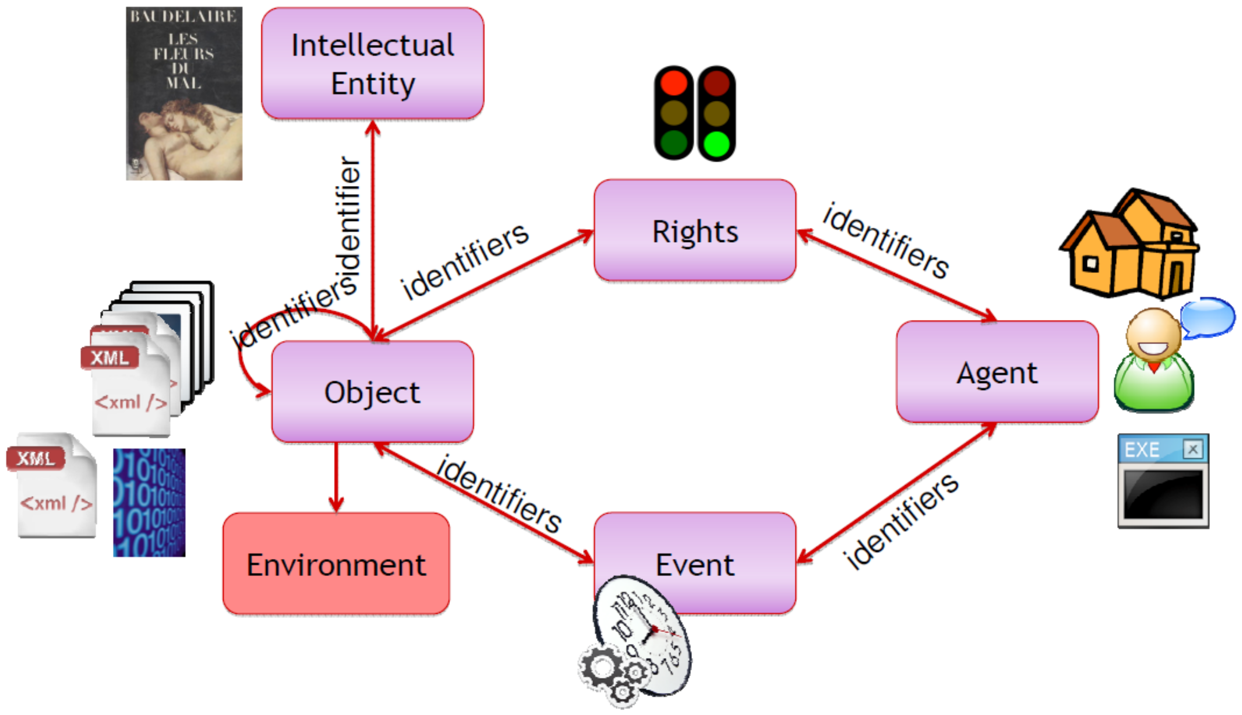


Figure 17: The PREMIS Data Model

Source: Sebastian Peyrard

Figure 18 below gives a brief example of what some of these metadata types might look like for a PDF file. In such a way, PREMIS therefore provides a framework of definitions to support capture of metadata relevant to the long term storage of data. PREMIS is implemented in several applications such as DAITSS.

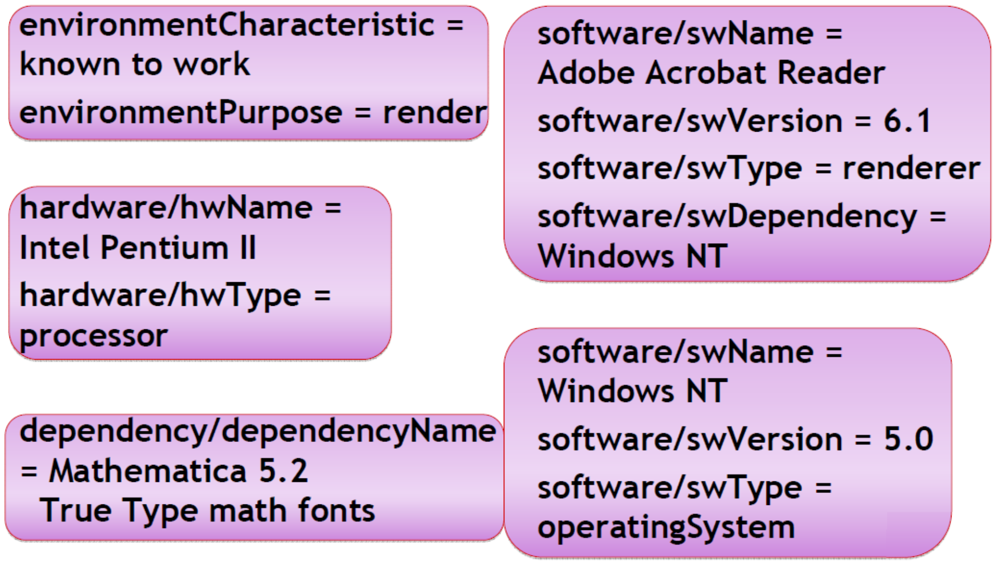

environmentCharacteristic = known to work
environmentPurpose = render
hardware/hwName = Intel Pentium II
hardware/hwType = processor
dependency/dependencyName = Mathematica 5.2 True Type math fonts
software/swName = Adobe Acrobat Reader
software/swVersion = 6.1
software/swType = renderer
software/swDependency = Windows NT
software/swName = Windows NT
software/swVersion = 5.0
software/swType = operatingSystem

Figure 18: PREMIS Environment Example

Source: POCOS conference 2012, A.Dappert, http://vimeo.com/36101909

A very good example of a PREMIS implementation is DAITSS (Dark Age in the Sun Shine State) available from the Florida Center for Library Automation (http://daitss.fcla.edu/). For additional information on PREMIS, please also see section 6.2.3 of TIMBUS deliverable D4.2.

3.2.4.1.1 PREMIS OWL

PREMIS OWL (website: http://www.loc.gov/standards/premis/owlOntology-announcement.html) is an implementation of the PREMIS data dictionary using the Web Ontology Language (OWL), instead of the XML schema commonly used for storing PREMIS metadata.

The usage of OWL has some benefits such as allowing bidirectional relations, but the framework is to be seen as a complement to the existing XML approach rather than a new development.

3.2.4.2 Dublin Core Metadata Initiative (DCMI)

The Dublin Core metadata initiative (http://dublincore.org/) was originally started in 1995 by the Online Computer Library Center (OCLC) consortium. Today it is an independent entity and is documented as a standard in RFC 5013, ISO 15836-2009 and NISO Z39.85. It is based on a set of 15 basic elements which are described in detail in the DCMI registry. Table 7 below provides a description of these 15 elements.

Table 7: Basic Elements of DCMI

Source: http://dcmi.kc.tsukuba.ac.jp/dcregistry/

Element Name	Description
Title	A name given to the resource
Creator	An entity primarily responsible for making the resource. Examples of a Creator include a person, an organization, or a service.
Subject	The topic of the resource. Typically, the subject will be represented using key-words, key phrases, or classification codes. Recommended best practice is to use a controlled vocabulary.
Description	An account of the resource. Description may include but is not limited to: an abstract, a table of contents, a graphical representation, or a free-text account of the resource
Contributor	An entity responsible for making contributions to the resource. Examples of a Contributor include a person, an organization, or a service
Date	A point or period of time associated with an event in the lifecycle of the re-source. Date may be used to express temporal information at any level of gran-ularity
Type	The nature or genre of the resource.
Format	The file format, physical medium, or dimensions of the resource.
Identifier	An unambiguous reference to the resource within a given context.
Source	A related resource from which the described resource is derived. The described resource may be derived from the related resource in whole or in part.
Language	A language of the resource
Relation	A related resource
Coverage	The spatial or temporal topic of the resource, the spatial applicability of the resource, or the jurisdiction under which the resource is relevant. Spatial topic and spatial applicability may be a named place or a location specified by its ge-ographic coordinates. Temporal topic may be a named period, date, or date range. A jurisdiction may be a named administrative entity or a geographic place to which the resource applies.
Rights	Information about rights held in and over the resource. Typically, rights infor-mation includes a statement about various property rights associated with the resource, including intellectual property rights.

Dublin Core is widely used by audio and video media owners and broadcasters today. Many implementations of the Dublin Core standard exist to provide a framework for content owners to capture metadata and store it with the media source. This approach is also used in national video archives.

3.3 Summary

This section presented relevant related work for the purpose of modelling context. The surveyed enterprise modelling frameworks show that several groups of stakeholders of a process can exist: those pertaining to the wider environment, such as regulatory stakeholders; those pertaining to the organization, such as the process owner; those pertaining to the process, such as an operator that interacts with a system that is used in a specific activity of a process. The different stakeholders of an organization's business process might have different concerns on what context should be relevant to maintain in the long-term. Those concerns depend on the viewpoint that the stakeholder takes on preservation.

In that sense, it can be assumed that the determination of the relevant context parameters is a task that is dependent on the specific case under analysis. A reference framework is an important tool to guide the effective determination of the relevant context parameters. The use of perspectives depicting different abstraction layers, which is common in such architecture frameworks, is an important tool for separating the different concerns and for partitioning the complexity of the task, with the most common setting being Business, Application and Technology. This can be observed in the cases of the Zachman Framework, TOGAF, and Archimate, which are described in section 3.1. Despite the inclusion of other perspectives, or the usage of other names although with the same meaning, those layers are effectively present.

Nevertheless, different perspectives are possible, which allow for decreasing the complexity for individual scenarios. As described in section 3.1.1, the Zachman framework can act as a classification grid for the types of objects that might exist in an organisation. Particularly, it offers further separation of concerns on the different perspectives of the organization by offering different perspectives or dimensions: *Data/What*, *Function/How*, *Network/Where*, *People/Who, Time/When*, and *Motivation/Why.* The usage of the Zachman Framework permits the classification of different conceptual parameters, ranging from legal requirements, i.e., Scope/Motivation cell down to actor roles, i.e., Business/People, or even software components, i.e., System/Function, and so on. Due to the holistic nature of the Zachman framework and its purpose, it can be used for aiding the identification and classification of the different contextual parameters relevant for preservation.

In order to model the context parameters of a business process and the dependencies (including semantics) between the two, different knowledge representation mechanisms can be used. However, there is a specific requirement for enabling reasoning on top of this knowledge, which will be explored in Task 4.2. This makes the usage of Ontologies appropriate for capturing this knowledge. Being a popular, standard, well performing (regarding reasoning), and expressive language for modelling certain knowledge (e.g., “all birds can fly”, as opposed to uncertain knowledge, e.g., “only nearly all birds can fly, as penguins cannot”), OWL comes out as a suitable option for this purpose.

4 Scenarios

This section presents relevant examples of scenarios identified by TIMBUS consortium members; they originate either from the working environment of the TIMBUS partners, or were identified in interviews and discussions with external parties.

These scenarios serve as illustrations of typical processes across various domains and businesses. A broad number of coarsely described scenarios has initially been identified, which formed the basis for developing the context model bottom up. The scenarios introduced below were afterwards described in much detail, and are employed to check whether the context model can adequately capture relevant aspects of the processes. It was thus aimed at obtaining a set of rather diverse scenarios, which would put different importance on certain contextual aspects, and thus complement each other in identifying different parts of the context model. We will focus here on a selected subset of scenarios that illustrate this diversity.

The descriptions follow this structure:

- Identification of use case
- Identification of stakeholders and their goals
- Identification of processes that enable/drive these goals
- Identification of relevant context parameters
- Identification of preservation requirements

The four scenarios include intellectual processes generating property rights, knowledge management, software escrow, and scientific data analysis via experiments. These scenarios vary from high-level process to focused and technical ones; from ones focusing on human factors to those rather focused on technical systems.

4.1 Intellectual property rights / patents

4.1.1 Use case

A primary driver behind business process preservation in the enterprise is legislative reasons. This scenario further outlines some of these drivers for digital preservation and the associated issues of addressing them. Deliverable D4.4 provides extensive information on this topic.

Commercial organisations provide services or products which are visible and tangible. However, the most valuable assets that any organisation possesses is its unique 'know-how', or knowledge assets. This 'know-how' distinguishes an organisation in the industry. Knowledge is therefore the most valuable asset an organisation possesses and its intangible nature can make it the most difficult to secure both physically and legally. In the case of long-lived products or patents, the industry is poorly equipped at present to meet possible future legal challenges.

Intellectual Property (IP) rights are rights on intangible assets like inventions, design or artistic works. Types of IP rights are patents, copyrights, trademarks and design rights. For a business it is important to have the exclusive rights on the IP belonging to it in order to prevent others from commercially exploiting it. The use case described in this section focuses on patents. Those grant the inventor the protection from others exploiting the IP for 20 years.

4.1.2 Stakeholders and their goals

The goals to be achieved when preserving processes concerning intellectual property are

- Developing preservation mechanisms to document the IP process and history to efficiently defend a position in court (lower cost, and that stand up to legal scrutiny). Currently, tools for this do not exist.
- Preventing yourself from redesigning products or pay license fees unnecessarily because you can't prove prior art for an existing product.
- Being able to render data which is in its original format (legal requirement today!)

4.1.3 Business Processes

Product Design consists of the steps (a) Exploration (b) Plan (c) Develop (d) Manufacture. This scenario will focus on the planning part of the process which is the most likely to produce IP. Figure 19 illustrates the main elements in the complete product design for a microprocessor; a high level description of the process can be found in (Abowd, 1999).

(1) At the conception phase, an idea is researched to identify if there is a valid market for a feature or product. The requirements of the feature/product are documented to meet the needs of the market and a design plan along with costing is put in place.

(2) If the business decides to proceed into a design phase, detailed design specifications are produced.

(3) In the development phase, these designs are prototyped. The new features are integrated with each other or into a pre-existing product. Any high volume manufacturing issues must also be resolved.

Presentations to support a marketing campaign based on the integration of features begin to be developed.

(4) At the manufacture phase, the product will go into high volume production which is supported by a marketing campaign.

Each step and each phase is recursive and issues found further along the chain may feed back into an earlier phase or step to be addressed. Each phase naturally also incurs additional investment and thus there is a business decision required whether to proceed or halt development. These decision points can be an ideal opportunity to archive the work to date in case it is of use in the future as shown in Figure 19.

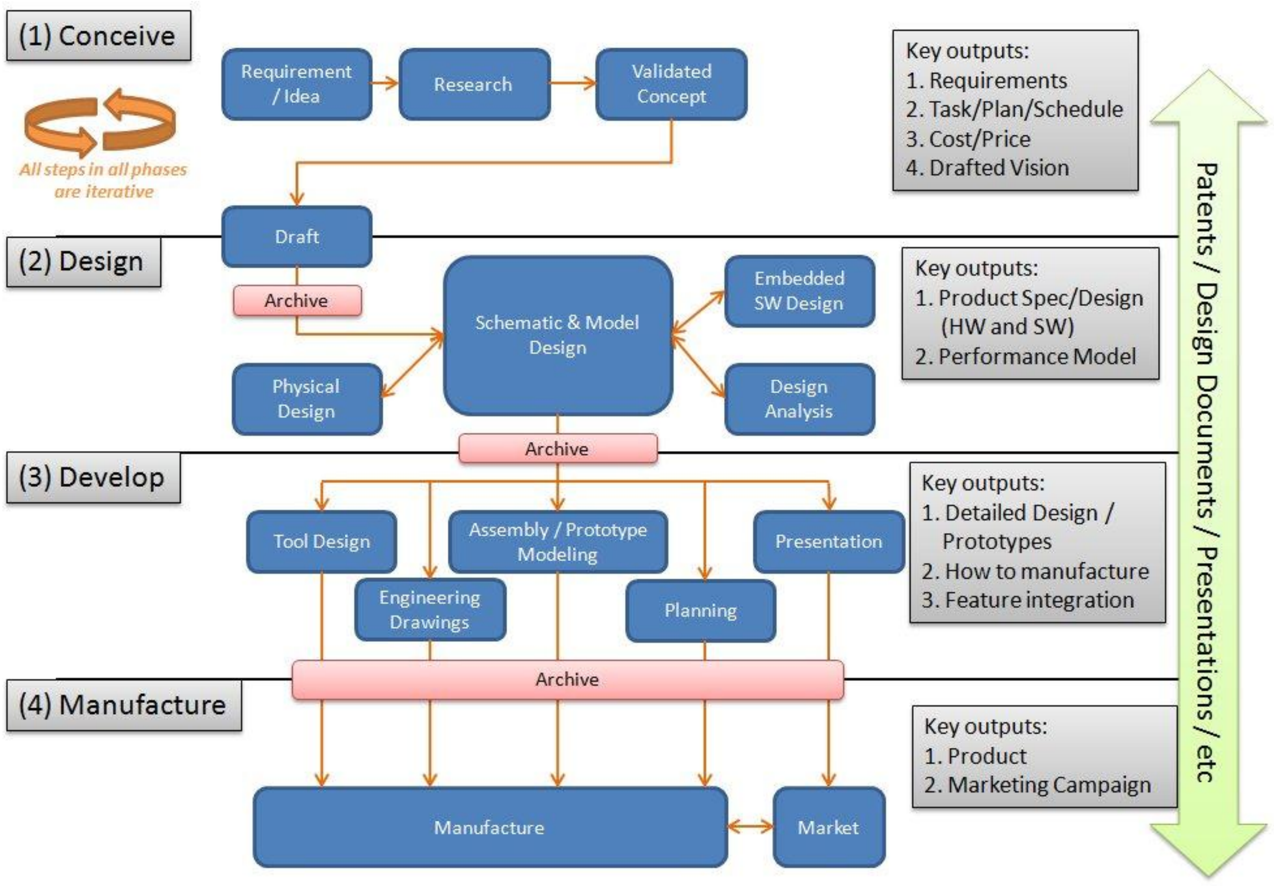


Figure 19: Intellectual Property Generation Process

4.1.3.1 Process Ontology

Important aspects of the process can be described in the model given Figure 20.

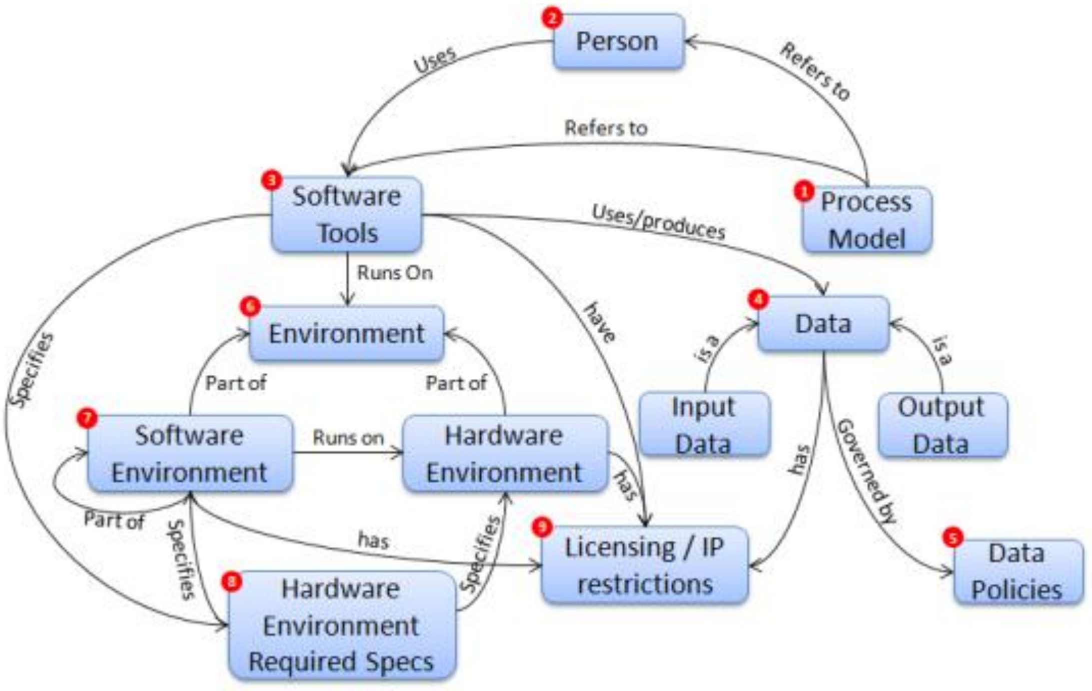


Figure 20: Intellectual Property Generation – Ontology

1. The process model defines who does what (via Person) and how they do it (via software tools).
2. People use software tools in a sequence defined by the process model. People provide "know-how" and possess implicit knowledge (potential to capture as context)
3. Software tools both use and produce data. They render and process data.
4. Data consists of input data to a software tool and output data once that software tool has processed the data
5. The data lifecycle is governed by data policies
6. Software tools run on environments; Environments consist of software environments (operating system, layered products, middleware) and hardware environments. In the PREMIS data dictionary, environment can also have other dependencies (for example, font files, XML schemas). But this ontology focuses on the ones most relevant to this scenario.
7. Software environments run on hardware environments. Software environments are also parts of other software environments (e.g.: a virtual machine is a Software environment itself but also is part of another software environment (i.e.: a hypervisor))
8. Software tools, software environments and hardware environments all specify particular hardware dependencies, or required specs. These can be externally monitored for obsolescence (for example, by an automated curation process)
9. SW tools, SW environment, HW environment and data may have licensing and/or IP restrictions

4.1.4 Relevant Context Parameters

Table 8 below shows which types of context parameters would be relevant to each of the ontology classes defined earlier in Figure 20.

Table 8: Relevant Context Parameters

Ontology Class	Relevant Context Parameters
Process Model	Defines a list of business processes, their constituent sub-tasks in order of execution and the people who must execute each step: BP name, task name, task execution order, task owner, required inputs, and expected outputs.
Person	Role, responsibilities, name, employee ID, skill set needed for role, usernames/passwords, etc
Software Tools	Tool name, tool version, tool owner, tool documentation (administrator, user, install and configuration documentation), required input, expected output, all upstream & downstream dependencies (can be dependent on other software or hardware elements).
Environment	Environment required to support a particular software tool; a collection of specified software packages installed and running in a specific operating system on a specific hardware environment. Environment should include: supported software tools and link to software, hardware environment and hardware environment required specification.
Software Environment	Operating system version. Installed patches, layered products and middleware that are not considered to be part of the software tools. Administrator, user, install and configuration documentation.
Hardware Environment	Generic information on CPU, RAM, disk, network, graphics, peripheral devices. No specific dependencies at this level.
Hardware Environment Required Specification	Specific dependencies on hardware devices/features required to support a specified environment.
Data	Size, format, list of owning/rendering applications, security/encryption inhibitors, etc.
Input Data	Data defined as the source data for a specific software tool
Output Data	Data defined as the output data from a specific software tool
Data Policies	Information lifecycle management policies, including data classification, how long to retain each data type and must be historically versioned as laws, classifications and other requirements will evolve over time.
Licensing/IP Restrictions	Licensing/IP information for specific software, hardware, software tools or data formats that governs access to data. Should include dates, IP/license owner, a copy of the IP/license agreement at that point in time, etc.

The work in this scenario has specified these more formally using the actual parameters from the Protégé model. The parameters described in the table are important as the particular steps, roles, individuals, inputs, outputs, and so on, taken in different design projects will change over time. In the future, this is valuable knowledge; some examples of where it would be useful is to help re-construct and re-run the process, or to prove the authenticity of the design in a litigation case, or figure out how a particular design issue occurred.

4.1.5 Digital Preservation requirements

We can identify a number of important requirements for preserving processes in the intellectual property domain.

- The specific time horizon of intellectual property has a couple of considerations:
    - The product patent lifetime is 20 years
    - The actual product lifetimes in service (e.g.: in aircraft, ships, tanks etc) could however be up to 80 years
    - Innovations can happen in the manufacturing process itself, and those might be retained for a much longer period than the actual product
- Exhumed tools have to run “like today”. Functional behaviour is more important than performance.
- The hardware environment must be exchangeable

4.2 Knowledge Management

4.2.1 Use Case

Knowledge management, as a scenario similar to the patent use case, was identified by iPharro and SAP. In particular, the knowledge creation and patent application processes at a large company are in focus. When a company like SAP is involved in a lawsuit about intellectual property it needs to prove that the IP originated in the company. Digital preservation should enable tracing back to the points of idea generation, submission and check for legal validity. This then lets you demonstrate which information systems supported the process of patent generation, what kind of information was available at the time and how it was accessible.

The knowledge creation and patent application processes are supported by an enterprise information portal (EIP) that includes a facility to submit innovations and to research for ideas previously submitted as part of an information lifecycle management approach. Furthermore, the EIP supports a federated search environment, i.e., information retrieval results in the portal are aggregated from different distributed search services. In this scenario, at least one of the contributing search engines is operated by a third party that provides a service that is instrumental for the EIP.

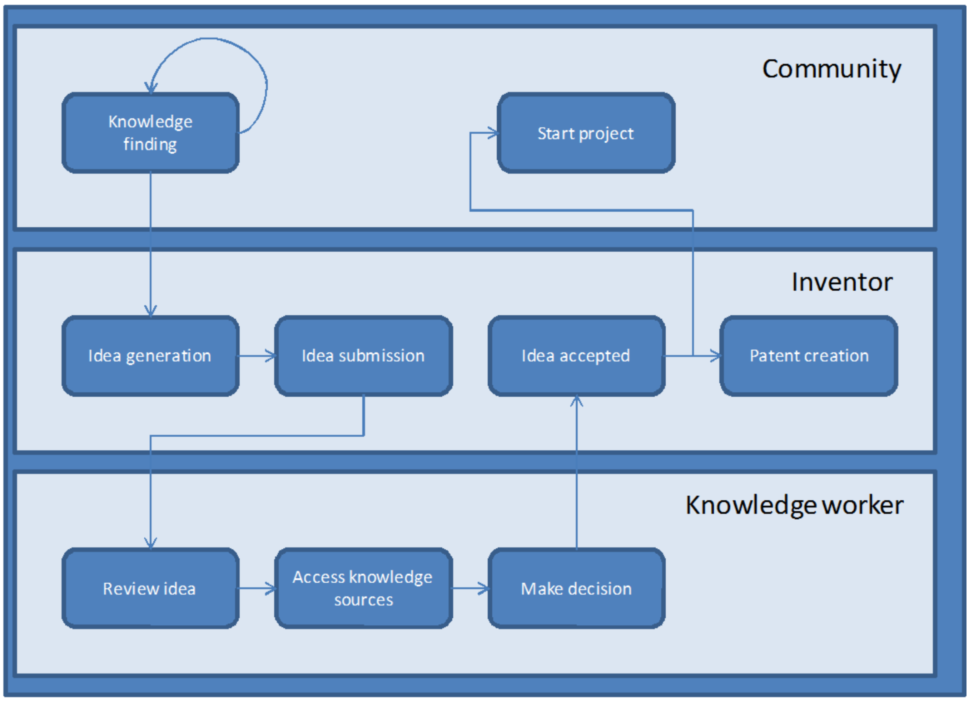


Figure 21: Knowledge Management Process

The inter-corporate relationship may be based on a Service Level Agreement (SLA) by which the service provider makes guarantees of information availability. In order to fulfil the SLA, the provider performs digital preservation of a relevant subset of its data served for the customer. Ideally, an interface is established between the preservation processes implemented in the portal and the external preservation process to determine which subset of index information needs preservation.

From the perspective of the service provider, preservation may furthermore be used as a form of business continuity management: Part of the SLA to its customer may be a downtime maximum in case of an emergency. Preservation of the distributed machinery used to support the federated search service may provide the functional basis of such guarantees.

4.2.2 Stakeholders and their goals

Although the scenario of knowledge management is similar to the previous one because support for patent claims is in focus, the stakeholders and goals vary because of its different structure. The principal stakeholders are the inventor (or group of inventors) that aims at obtaining benefits for an innovation, and reviewers of the idea: a knowledge worker and at later stages a patent lawyer who manages patent application processes and aims at creating them as defendable as possible in court while keeping IPR costs low. In addition, other members of the community play a role as creators of media that are used as dependent information in the patent application process. This may include media about basis technologies or scenario information.

4.2.3 Process model

In SAP knowledge is organised in various formats. The NetWeaver Portal offers a collection of news regarding the company, internal guidelines and also knowledge created by employees in various formats. Those formats include wiki pages, text, images, videos of talks where ideas are presented. An overview of the general knowledge management process is given in Figure 21, which includes idea generation and patent preparation. Idea generation follows a "spaghetti" process, one that is not formally modelled even though it is supported by the various information systems mentioned[7]. When an employee decides to move forward with an idea, one example of doing this is provided by an idea management system. Here the employee can enter an idea, describe it and its benefits for the company and how it could be implemented. The idea is then assigned to a knowledge worker. This process is very structured and supported by the idea management system, as shown in Figure 22. After assignment the knowledge worker has to review it and make a decision whether to accept it or not. The process of reviewing is again unstructured and has a high variance. The knowledge worker has to access various sources of internal and external information to evaluate the novelty of the idea. Those sources can include existing patents and the knowledge stored in the company portal mentioned above. An accepted idea could then be followed by an implementation, the start of an internal project or the filing of a patent. In this patent process an employee (inventor) files an Invention Disclosure Form (IDF, as shown in the portal page in Figure 23).

[7] A "spaghetti" process is the opposite of a more formally structured "lasagna" process, according to a terminology formulated by W. van der Aalst (van der Aalst, 2011).

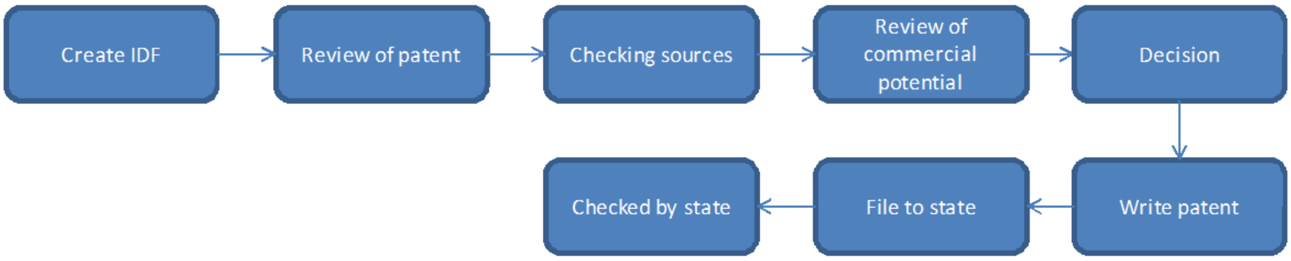
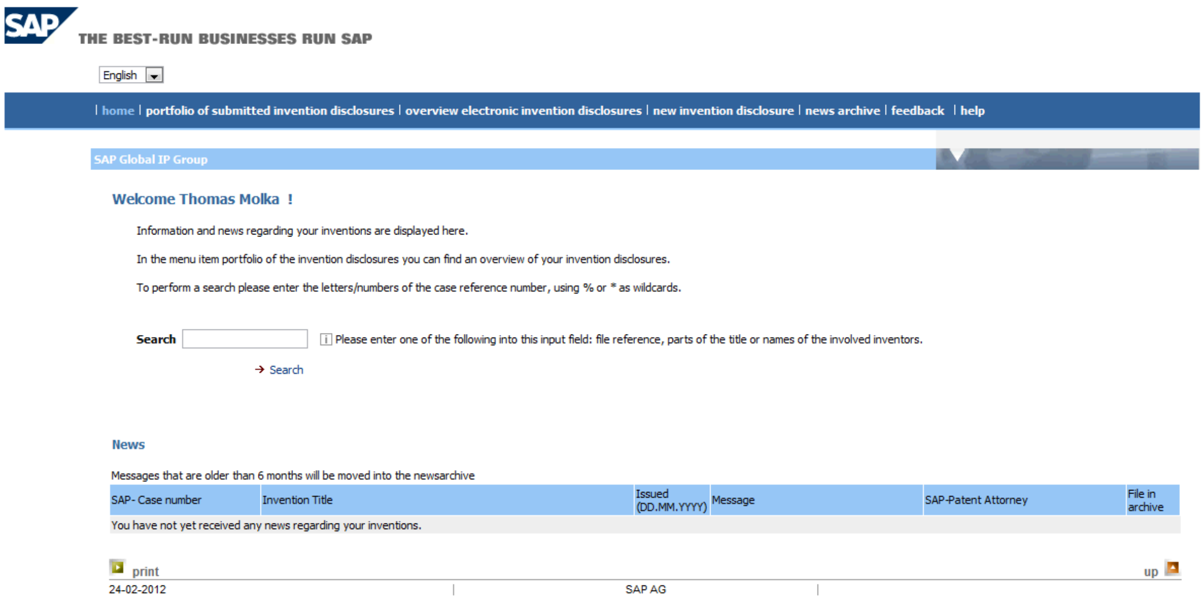


Figure 22: Patent Review

This initiates the review where a patent lawyer is assigned to review the IDF. Again there is a more thorough check of internal and external sources to examine if there are potential legal problems due to existing IP. This, as in the knowledge worker's process of reviewing ideas, will be supported by iPharro software enabling search across different types of media as part of the federated portal search functionality.[8]



THE BEST-RUN BUSINESSES RUN SAP

English

| home | portfolio of submitted invention disclosures | overview electronic invention disclosures | new invention disclosure | news archive | feedback | help

SAP Global IP Group

Welcome Thomas Molka !

Information and news regarding your inventions are displayed here.

In the menu item portfolio of the invention disclosures you can find an overview of your invention disclosures.

To perform a search please enter the letters/numbers of the case reference number, using % or * as wildcards.

Search — Please enter one of the following into this input field: file reference, parts of the title or names of the involved inventors.

→ Search

News

Messages that are older than 6 months will be moved into the newsarchive

SAP- Case number	Invention Title	Issued (DD.MM.YYYY)	Message	SAP-Patent Attorney	File in archive
You have not yet received any news regarding your inventions.					

print — up

24-02-2012 | SAP AG |

Figure 23: Patent Submission

If a decision for filing the patent is taken, the patent is written and a formal application initiated, which is reviewed by the respective patent office.

The indexing and retrieval service that the provider (here: iPharro) performs is an important example of inter-corporate context: In the knowledge management process and in particular the IPR protection process, the service provider needs to ensure that the information about the patent application process are kept consistent and accessible over the complete duration of the patent application process (which often lasts for five to ten years) and in most cases also its granting period to be able to defend the patent before court (which adds another 20 years). An additional requirement is to preserve the state of the index at a given point in time so each review step can be re-enacted.

[8] Federated search performs retrieval over multiple search engines.

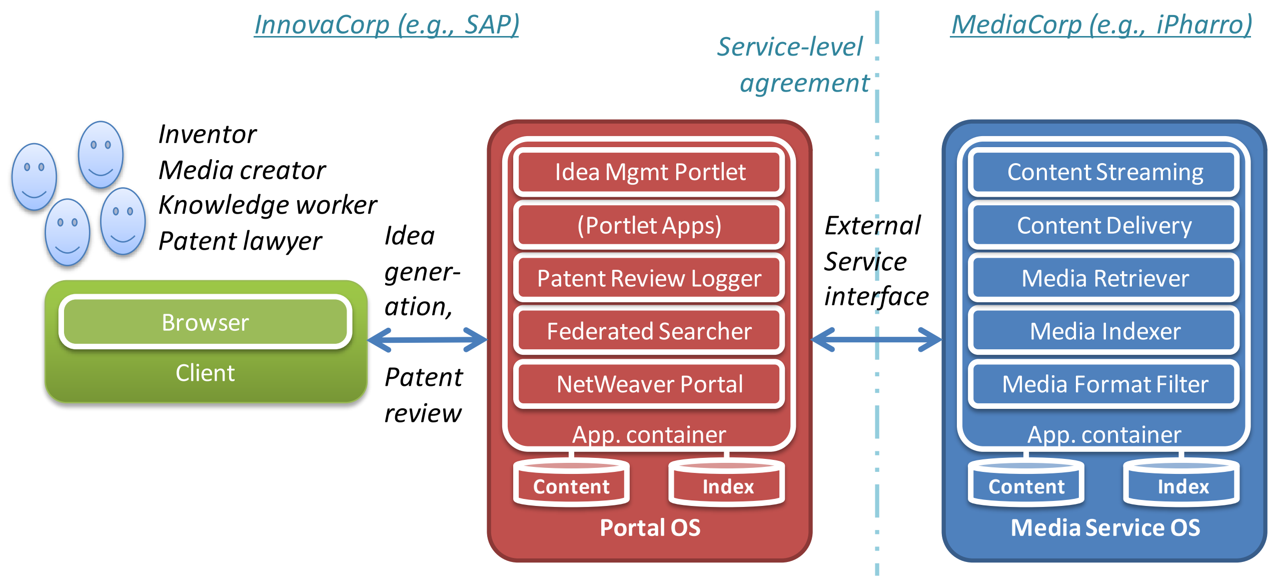


Figure 24: System overview of knowledge management portal.

An overview of the high-level system is shown in Figure 24 where InnovaCorp represents a large corporation like SAP and MediaCorp a service provider like iPharro. Stakeholders use the system to manage their information using the NetWeaver portal in the center. As one of the portal applications, the idea management portlet is used, first from the perspective of the inventor when filling the IDF. As described above, this patent review process is executed by the patent lawyer and represents the actual process to be identified and preserved. For this, a business process logging system is used that allows to re-enact the un-structured review process that consists, among others, of search requests run on the federated search engine. This search engine, in turn, depends on the retrieved results of the external media content provider that offers advanced media indexing and streaming services.

4.2.4 Relevant Context Parameters

Table 9 gives an overview of the most important context objects and parameters that are relevant for preservation of the knowledge management process, in particular the patent review process. Their identification was mainly guided by the question what is necessary to illustrate what information was available at a certain point in time.[9] In this table, italicised concepts like *document* have generic parameters that are explained at the bottom.

[9] The format of this table has been varied to capture the higher complexity of the scenario. The resolution into an ontology instance that connects the concepts is done in section 6.3.2.

Table 9: Relevant Context Parameters

Scenario aspect	Context objects and relevant parameters
Business strategy (Why + who, scope perspective)	Service-level agreement (SLA): contractual parties (service consumer InnovaCorp and provider MediaCorp), specification (*documents*), confidentiality, duration, generic terms of service (*documents*) Corporate documentation of IPR and patenting policy and connected document retention rules (*documents*, InnovaCorp) Service documentation: generic provisions (*documents,* configuration *data*, e.g., for WebService interaction; MediaCorp)
Process models (How, business perspective)	Process model for the overall process and sub-processes: process structure captured in BPNM or other meta-model (*document*) Causal network (van der Aalst, Adriansyah and van Dongen, 2011) / Markov model (Rabiner and Juang, 1986) for the unstructured processes (idea generation, review processes): process structure captured in some serialisable representation (*document*) Note that the actual business process instances are of interest, rather than a generic representation. This is why information in the models is specific, naming actors, their actions and the dependent documents.
Software, Systems (How + where, system level)	NetWeaver Portal (InnovaCorp): software including idea management system and patent review logging agent implementation (*software,* configuration *data*) External software (InnovaCorp): browser, file format plugins (*software, configuration data*) Media service system (MediaCorp): software for indexing, retrieval, content delivery and streaming system, based on operating system and service container (*software, configuration data*)
Roles (Who, business perspective)	Person (InnovaCorp): inventor (creates ideas), patent lawyer (or knowledge worker; processes ideas), media creator in knowledge-base (source of dependent information) ComputerAgent: media indexer (MediaCorp; indexes media files), patent review logger (InnovaCorp; extracts structure of review process) (*software, configuration data*)
Data/Knowledge artefacts (What, scope to technology perspectives)	Structured artefacts: patent (*document*s, protects IPR of idea), idea (*document*s, generic artefact that is turned into patent, used for product), IDF (*document*, belongs to idea) Unstructured media content (*documents, media files*, connected to idea as dependent documents or search results) Index data (allow retrieval): like *document*, operational parameters (indexing/search engine-specific, e.g., slice information for sub-indices or differential snapshots)
Data format instances (What + how, components perspective)	Standard file *formats*: Text (with encoding), Office (PPT, PPTX, XLS, XLSX, DOC, DOCX, etc.), PDF, Audio (MP3, AAC, WAV, AIFF), Video (AVI, FLV, F4V, MOV; Codecs: H.264, VC1, VP6, WebM, MPEG2, etc.). Specialised *formats* and containers: IDF document, patent application bundle, Webcast presentation (slides + speaker videos + meta-data)
Patent Lifecycle (When, business perspective)	Idea development phase, idea review phase, patent application phase, term of patent

Generic data concepts	*Data:* name, *format*, version, date, size, address (URI, DOI), access rights *Document*: like *data*, additionally: authors (*persons*), dependent *documents* (linked, referenced) *Format*: version, specification, encoding information, decoder/viewer application (*software*), encryption information, decryption application (*software*, mostly identical with viewer, e.g., PDF Reader) *Media file*: like *document*, additionally: duration, codecs (*software*) *Software*: name, version, specification/documentation (*documents*), software dependencies (*software*), data dependencies (*data*), configuration parameters (*data*), environment dependencies (operating system, virtual or physical hardware; see section 4.1.4)

4.2.5 Digital Preservation requirements

The main requirements of the scenario are as follows:

- Time frame for preservation: Patent application duration: 5-10 years + 20 years patent granting period.
- Between InnovaCorp and MediaCorp, an SLA is necessary, both for the service itself and as a consequence of the internal preservation requirement also the future availability of preserved versions. The SLA should support information exchange about preservation of objects and the corresponding indexing information.
- For business continuity management, the systems are made restorable with a maximum down-time, so exhumation must be possible with low effort.
- The exhumed tools must reflect the functional behaviour of selected features at a given point in time

4.3 Software Escrow

4.3.1 Use Case

Software Escrow addresses the well-established concept of outsourcing that aims to "sub-contract responsibility for all or part of an IT function to a third-party service provider that managed and operates the work". Today, over 7% of all IT-budgets are spent towards outsourcing contracts and this ratio will – accordingly to analyses by Gartner (Gartner,2008)– increase dramatically to 25% for 2020. Interestingly enough the current hype of cloud computing is one specific type of outsourcing and will account for 70% of the overall outsourcing budgets in 2020.

Software development and software customization are often made by external resources. The fundamental concept for all outsourcing contracts is to delegate responsibility (and risks) to a third party. The advantages of doing so are obvious:

- focussing on core business
- specialisation for the service provider
- improved explication of process planning and budgeting

On the other hand, delegating responsibilities introduces new risks as an undesired side effect:

- purchasers become dependent on external provider
    - outsourcing provider may go out of business
    - outsourcing provider may be acquired by a purchasers competitor
    - outsourcing provider may buy a competitor of the purchaser
    - purchaser cannot control the pricing

The aforementioned simplified risks are addressed by Software Escrow agreements. The hope is that Software Escrow reduces at least the impacts of the described risks.

An additional need comes from the DP community itself. The lifetime of a Software product is in general shorter than the time, digital artefacts will be archived. The risk, Software Escrow can mitigate is that the software vendor goes out of business or is no longer interested in the development of the needed Software. On the other hand, software escrow should protect the interests of a software vendor.

4.3.2 Stakeholders and their goals

A software escrow is a three-party arrangement: "*An independent trustee – usually a firm in the business of doing technology escrows – is appointed as the escrow agent for licensor and licensee. The parties enter into a three-way agreement. The licensor delivers a copy of the source code to the escrow agent, and is usually required to deliver a source code update whenever it delivers a corresponding object code update to the licensee under the corresponding license agreement. Upon occurrence of a triggering event, and only then, the escrow agent delivers the escrowed source code to the licensee*" (Meeker, 2003)

The risk mitigation approach is as follows: The software purchaser and the provider maintain their legal status and the level of information exchanged between both stays unchanged. The purchaser receives the binary parts of the software needed to execute the product like before – without any change of IP rights. In daily business the role of the trustee does not affect the IP discussion. He receives all information (such as the source code) solely to file away. However, if a so called escrow clause is triggered (e.g. if the supplier goes out of business), and only then, the trustee hands out all deposited information with the goal to enable the licensee to continue operation and maintenance of the licensed application.

This type of service is well established in today's IT market. Market leader NCC for example reports a revenue of 17,9m£ only in the UK with over 100 FTEs (NCC Group plc, 2010). Behind that, software escrow is often requested in software development for the public sector.

4.3.3 Process to preserve

The main criticism on the classical escrow is the focus on source code artefacts and in the maximum source code collaterals. Opposite to that, the improved holistic escrow addresses the preservation of all needed artefacts to beware the development process. It is obvious that a software development process consists of more than code. In a generic understanding, a software development process consists of:

- requirements analysis
- architectural design
- detailed design
- coding and testing
- integration
- installation
- acceptance support.

These processes are very strong oriented on the system to develop. Beside that some more administrative stuff is needed. To have a more concrete example, the V-Model 97 consists of 4 sub models (synonymous for sub processes). These are software engineering, quality assurance, project management and configuration management.

A short description of the successor of the V-Modell97, the V-Model XT, will make the picture of typical processes in a software development project a bit clearer. A description of the V-Model XT can be found at http://v-modell.iabg.de/v-modell-xt-html-english. It illustrates very well, that a software development project (and within the development process) consists of more than an only source code. The product index lists 110 kinds of products (or artefacts) which are developed and used during a software development project. Every project has to tailor the V-Model in an initial phase, so it is not possible to have a general description of the process for all projects and nevertheless for the preservation of a software project. However it is a very case specific and context dependent decision, what artefacts are needed to describe the process good enough, to fulfil the preservation needs.

4.3.4 Relevant Context Parameters

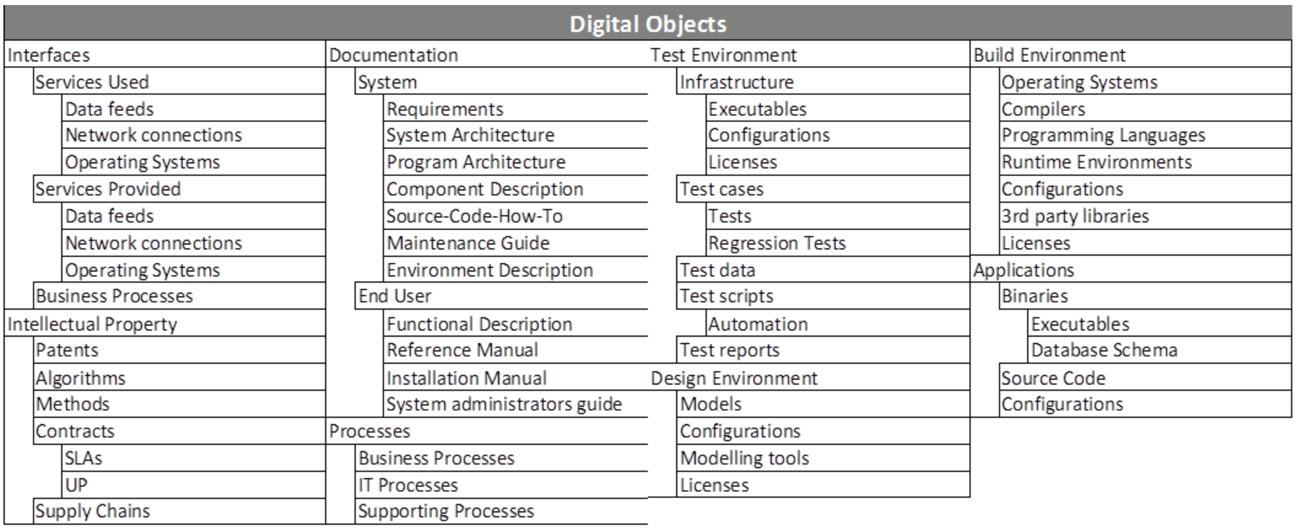
The goal of a software development process is to develop and in the end to deliver a product in an adequate quality. Beginning with the ISO 25010, the context of a product was introduced as an important aspect for the assessment of the different quality characteristics. The different quality characteristics defined by the ISO 25010 are:

- functional suitability
- performance efficiency
- compatibility
- usability
- reliability
- security
- maintainability
- portability.

For the assessment of a software project/product with a strong focus on preserving it for the future (or a for different development environment) the characteristics are very helpful for measuring the quality. The concept of adopting the characteristics to different digital objects is well established in the Quality risk management (QRM), which defines a practical framework for measuring and managing the quality of software products.

One aspect that is very helpful for preserving software in an Escrow is the fact, that some aspects of preservation (e.g. maintainability, portability) can be directly defined as focussed quality characteristics and the product can within be optimized for being preservable.

A possible initial taxonomy of preservable objects (and within context) is illustrated in Figure 25.


Digital Objects

- Interfaces
  - Services Used
    - Data feeds
    - Network connections
    - Operating Systems
  - Services Provided
    - Data feeds
    - Network connections
    - Operating Systems
  - Business Processes
- Intellectual Property
  - Patents
  - Algorithms
  - Methods
  - Contracts
    - SLAs
    - UP
  - Supply Chains
- Documentation
  - System
    - Requirements
    - System Architecture
    - Program Architecture
    - Component Description
    - Source-Code-How-To
    - Maintenance Guide
    - Environment Description
  - End User
    - Functional Description
    - Reference Manual
    - Installation Manual
    - System administrators guide
- Processes
  - Business Processes
  - IT Processes
  - Supporting Processes
- Test Environment
  - Infrastructure
    - Executables
    - Configurations
    - Licenses
  - Test cases
    - Tests
    - Regression Tests
  - Test data
  - Test scripts
    - Automation
  - Test reports
- Design Environment
  - Models
  - Configurations
  - Modelling tools
  - Licenses
- Build Environment
  - Operating Systems
  - Compilers
  - Programming Languages
  - Runtime Environments
  - Configurations
  - 3rd party libraries
  - Licenses
- Applications
  - Binaries
    - Executables
    - Database Schema
  - Source Code
  - Configurations

Figure 25: Digital Objects for Software Escrow

Every object can have different attributes which has to be in a well-defined and specific state for preserving the object. All attributes within an object can be measured and aggregated within special control points derived from indicators.

A more complex challenge is the preservation of tacit knowledge. It's obvious, that some knowledge is bounded to heads, especially while software engineering is just a creative process. That implies that some knowledge about a software development project will get lost every time. But some parts can also be preserved, if their potential lost is identified. This means especially things like standards or best practices. For example the definition of UML (or the location where it can be found) will be no longer of relevance for the daily business and will get lost at some day. The potential lost of this knowledge could be indicated in a very early stage and suitable tasks could be initiated.

4.3.5 Legal Aspects

As software is mainly an intellectual product, legal aspects are very important in preserving software development processes. There are questions like IP rights on Source Code, reselling of needed tools and more. The entire question is discussed more detailed in a section above in D4.4. It is planned to have some more specialised research and analysis of adopting legal aspects in the 2nd year of the TIMBUS project.

4.4 Scientific Data Analysis/Scientific Experiment

4.4.1 Use Case

This use case stems from the domain of data analysis. Specifically, the scenario setting is dealing with conducting a scientific experiment in the area of information retrieval and machine learning, where we want to verify and validate the usefulness of a method for assigning meta-data to items in a specific dataset. The concrete task in this use case is the automatic classification of items in a music collection into a set of predefined categories such as genre labels-

This scenario is chosen for several reasons. First, it is similar in its nature to the eScience setting in work package 9. Further, the process is interesting as it involves several different, partly remotely located data sources, services and tools. Further, the partners have access to all relevant parts of the process components as they are open-source and publicly available. Finally, one partner has an in-depth prior knowledge in running the process, from a former employment at the Technical University of Vienna.

4.4.2 Stakeholders and their goals

The primary stakeholder in this use case is the researcher who developed a method for automatic classification of music objects. The researcher performs a set of scientific experiments to test the method for its validity and usefulness, which can be shown by achieving a result, measured on a certain performance metric, which would be comparable to the current state of the art in the field.

A motivation for preserving this process is for example a scenario where a reviewer or other researcher in the field is challenging the results, claiming that the scientific work was not sound, and thus the results were not valid. This could be damaging for the researchers reputation in their research community. Thus the researcher would like to prove that the underlying research was according to best practice standards. In such a case, having the process of the experiments preserved, the researcher has the chance of exhuming the whole process and to prove that the experiment was conducted according to best practices at the time.

Another motivation for preserving such a process is in the context of executable papers, which aim at enhancing publications that are data centric with the ability for the reader to rerun the experiments, potentially under slightly modified parameters.

4.4.3 Processes

When performing automatic genre classification of a music collection, a process typically consists of (some of) the following steps:

- Acquiring a set of training and test data. This data set can potentially be provided by remote content providers, e.g. online stores such as Amazon.com, 7digital.com, or others. Those might provide a web service or another specific protocol to acquire the sample data.
- Further, meta-data, such as a categorisation with genre labels, might be acquired from the same or a different data source, such as the All Music guide (http://www.allmusic.com/), Gracenote (http://www.gracenote.com/), or MusicBrainz (http://musicbrainz.org/). This assigned value might

be user generated data (e.g. via tagging), and thus change over time (e.g. from alternative rock to a later-emerging sub-genre of grunge)

- Pre-processing of the input data, for example format conversion from MP3 to raw audio, selection of relevant parts of the data (middle segments in music, specific paragraphs of a text, ...)
- Extraction of representative, numerical features from the input data. There is a plethora of remote services emerging, for example The Echo Nest (http://the.echonest.com). Tools for extraction might also be used offline on the same machine, but might come in different languages (C++, Java, Matlab), depending on different third-party libraries. There is potentially a timing aspect regarding the services used. Remote services might change frequently, and extract new types of representations, or extract existing representations in a different way, thus providing different numerical values that might significantly change the outcome. Local implementations might utilise some of the computation algorithms provided with the specific language, such as the Fourier transform by Matlab. These might differ over different languages.
- Storing of the features in some way (text, database) and format (e.g. WEKA ARFF (WEKA, 2012))
- Using a machine learning toolkit to train a model and assign new meta-data (genre labels) to unknown data. Different versions might provide different implementations of algorithms, and can thus lead to different results.
- Finally, the results obtained are presented, for example in the form of a publication

This process can be modelled in BPMN notation as illustrated in Figure 26:

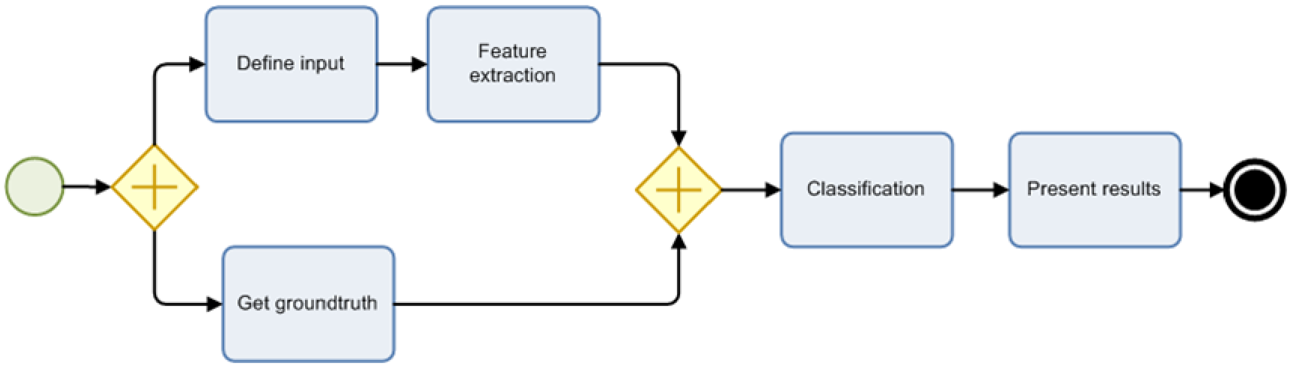


Figure 26: BPMN model of scientific experiment

4.4.3.1 Detailed analysis of the process

We modelled a specific instance of the scientific experiment process in the Taverna Workflow engine[10]. A model of the workflow can be seen Figure 26.

[10] http://www.taverna.org.uk

In the figure, "brown" boxes define the Workflow engine specific scripts; purple boxes are scripts based on predefined operations, such as Base64 encoding. Grey boxes define static input data, while the cyan boxes are the process outputs.

Modelling the process in such an engine is already a migration of the process; it has the side-effect of becoming more platform independent, as this workflow execution depends on the workflow engine, and not any more directly on other software environments. In this specific case, steps that might normally be performed by shell comands/scripts are replaced by a specific script-language known to Taverna. Also, all software components that are used have to be understood by the workflow engine, which thus becomes a layer of abstraction from the underlying operating system.

In this process, of the greatest importance to preserve and rerun at a later time is the technical system, and the data it produces. Thus, these two aspects are described in detail.

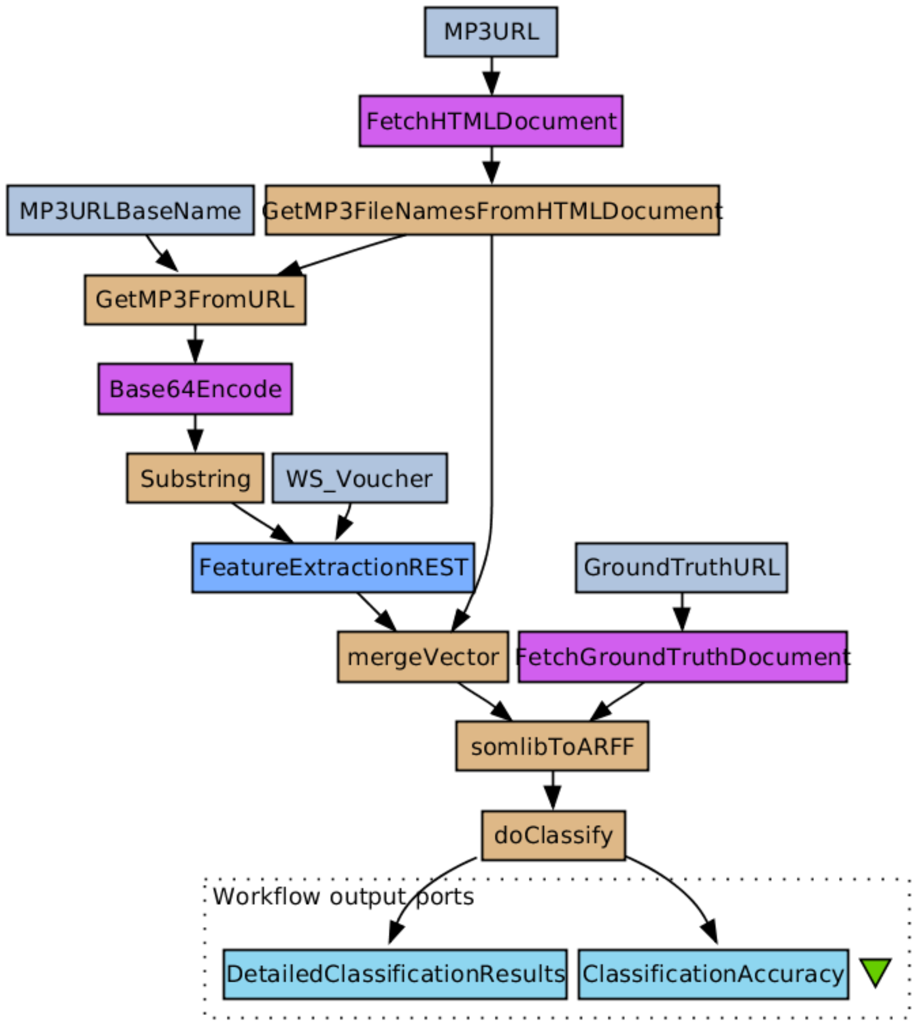


Figure 27: Taverna Workflow view of music classification process

The workflow illustrated in Figure 27 above consists of and depends on the following software libraries, systems and (external) services:

- WEKA machine learning toolkit, version 3.6.6; employed for the learning of a predictive model and assigning of labels to unknown data
- Java SOMToolbox, version 0.7.5.1; used for format conversions
- Taverna Workflow Engine, version 2.3.0; used to execute beanshell scripts and to provide the process workflow
    - Taverna requires Graphviz for rendering the workflow chain
- Java Development Kit / Java Runtime Environment version 6.0; use as runtime environment for the Taverna Workflow Engineering
- Ubuntu Linux version 11.04; used as platform to run the JDK / JRE
- AudioFeatureExtraction REST Service, running at http://kronos.ifs.tuwien.ac.at:8080/fex/featureExtractionREST; provides the extraction of numerical features from MP3
    - CGI parameters:
        - voucher={authentication key}
        - music={mp3 file Base64 encoded}
    - Return value: Vector in SOMLib format
- MP3 Data provider Service; provides the audio files. For demonstration purposes, this is a simple Apache (version 2.2.0) directory listing, accessible at http://kronos.ifs.tuwien.ac.at/timbus/musicProcess/music/
- Genre assignment (ground truth) provider; provides the assignment of the audio files to a specific genre. For demonstration purposes, modelled as a simple HTTP service, available at http://kronos.ifs.tuwien.ac.at/timbus/musicProcess/genres.txt, in SOMLib format.

The process is very data intensive, producing a lot of data that is produced by one process step and consumed or transformed by subsequent ones. In detail, the single steps have the following data associated with:

- Fetch data from data provider
    - Output: source data, e.g. in MP3 Format
- Store and prepare data
    - Input: source data
    - Output: prepared data, potentially migrated to different format and with other pre-processing applied
- Feature extraction
    - Input: prepared data

- Output: numeric representation ("features")

- Fetch genre / ground truth
    - Input: e.g. artist and song name, track-ID...
    - Output: genre (or other category label)
- Data combination
    - Input: feature file
    - Input: genre information
    - Output: combined file in e.g. WEKA ARFF format
- Learning
    - Input: combined file
    - Input: learning parameters
    - Output: Learning Results, generally a real number obtained from some metric

4.4.4 Relevant Context Parameters

Table 10 below shows which types of context parameters are relevant to this scenario.

Table 10: Relevant Context Parameters

Ontology Class	Relevant Context Parameters
Model	BPMN Business Process Model Taverna Workflow Model
Persons	Role, responsibilities, skill set needed for researchers
Agents	Operators of different services (data, groundtruth and feature extraction provider) , their motivation and goals for operating the service
Software Tools	Tool name, version, owner, documentation, input and output parameters of the tools described above
Software Environment	Operating system version, and installed versions of execution platform (Java Virtual Machine in this specific case)
Data	Process input and output data; data produced and consumed inside the process, as described above

4.4.5 Digital Preservation requirements

A primary motivation identified for preserving this process is for settings where the results obtained are questioned. In such a case, it is important to have the preserved system behave equal to the system of today – a slight deviation will be noticeable from a different output value, and then the repeatability of the whole process is questionable.

The time horizon for which the preservation of the system is required cannot be determined as an absolute value, just in relation to other conditions. One assumption would be that it mostly correlates to the time horizon during which the results obtained are relevant to the research community, with the assumption that they become less relevant once the method has been superseded by others yielding better results.

Challenges in preserving the process go beyond preserving the software components utilized alone. In particular, the use of external services adds requirements that these services are still available in the future, and are delivering the same results. Feature extraction services are very volatile in their nature, and the type of information they compute, and how they compute it might change without prior notice. Also, the business models behind these services are not entirely clear, thus sustainability of the services can not be guaranteed. Finally, some services might not change in functionality and availability, but in the data they provide. As such, the categorization of music into genres might not be a static function, but might change over time with the emergence of new genres. Also, the data available by the data providers is volatile – the music provided might change according to sales and popularity, and format and quality of the encoding is subject to changes as well.

All these aspects have direct impact on the preservability of the process. To verify the correct functioning of the individual components, capturing of the data communication flow between them seems critical.

5 Context Parameters

This section contains concise descriptions and informal definitions of the relevant context parameters derived from the earlier illustrated scenarios in section 4, and refined according to the related work described in section 3. The extracted context descriptions are meant to be an informal, but precise. It should serve as a dictionary of relevant context parameters that is referenced by the Context Model and Context Model Instances which are illustrated in section 6 in more detail.

We can distinguish several types of context parameters valuable for preservation depending on the viewpoint of a specific stakeholder. Context parameters are important to some stakeholders and not to others. Depending on the concerns of a stakeholder, their concerns will be at one (e.g., business) or multiple abstraction levels (e.g. business, strategy). For instance, if we focus on a strategic direction, we can say that an (operational) goal can be assessed and contributes to the vision of the organization as a means for fulfilling strategic goals. However, the achievement of goals is influenced by the external rules (e.g. legal requirements) and also by internal rules (e.g. operating procedures, compliance with standards). If we focus on a technological direction, information systems comprise components and coordinate services that are used to support business processes, and so on.

Based on that rationale, the following subsections present different perspectives based on the Zachman Framework on the context parameters derived from the scenarios, which were divided according to the abstraction layer they belong to, so that separation of concerns is achieved. The first subsection (Scope/Context) adopts the perspective reflected in the top row of the Zachman framework, which is related to the business context surrounding the organization. In other words we look to the outside of the organization and to its interfaces with the exterior. The second subsection (Business/Conceptual) adopts the perspective reflected in the Business row of the Zachman framework, which describes the internal workings and structure of the organization. The third subsection (System/Logical) adopts the perspective reflected on the System row (the third counting from the top) of the Zachman Framework, which describes how the systems of the organization will satisfy the organization’s information needs. Finally, the fourth subsection (Technology/Physical) adopts the perspective reflected in the Technology row of the Zachman framework, which describes implementation aspects of systems.

5.1 Scope/Context

As already referred, the parameters existing at the level of the Scope/Context reflect things existing outside the “walls” of the organization or things that belong to the organization but that are “visible” from the outside. Looking at the top row in Zachman and at then at the different columns, examples of those things can be data or resources important to the business (i.e., the data column), functions that the organization possesses and that are used in interactions with the outside world (i.e., the function column), locations important to the business (i.e., the network column), other competitor organizations (i.e., the people column), relevant business events (i.e., the time column), or even drivers and constraints to the business (i.e., the motivation column). Table 11 describes relevant context parameters belonging to the Context/Scope perspective.

Table 11: Context/Scope Parameters

Parameter	**Description**
External Resource	From a transformational standpoint, business processes use resources as inputs and outputs. Resources can be tangible or intangible and range from materials to information (examples of resources include the packages involved on a transportation business process, the bar code on a package (as an optical machine-readable representation of data), the actual data that is physically represented by the bar code, etc.). In this case this parameter represents resources that are outside the boundaries of the organization (i.e., of which the organization has no ownership), but that are important.
Business Service	A business service is a capability offered by an organization through an interface to the exterior.
Geo Location	A place where the activities of an organization take place.
Business Entity	An external entity important to the organization.
Time Zone	A region where the same standard time is adopted.
External Event	An external occurrence that affects the organization.
Driver	A condition that influences the setting of goals by the organization.
Constraint	A condition that prevents the organization from setting determined goals.

Each of those parameters can be further broken down as required. For instance, one important aspect which functions as a contextual driver or constraint to the business is the legal aspect. Legal aspects are important to TIMBUS since they can affect the whole operation of the organization, and with that, other context parameters. Although the dependencies existing between legal context parameters and other parameters are not explicit at a first glance, they become explicit, for instance, when dealing with the preservation of software systems under a commercial license or with data under copyright. Table 12 defines legal context parameters that assume particular importance in TIMBUS.

Table 12: Legal Context Parameters

Parameter	**Description**
Contract	Legally binding agreement
Service Level Agreement (SLA)	Agreement between customer and provider determining in detail the services that should be delivered (and consequences in the event of not delivering as agreed)
License	Formal authority to do something that would otherwise be unlawful
Escrow Agreement	A written agreement between two or more parties whereby the grantor, promisor or obligor, delivers certain instruments or property into the hands of a third party, the escrow agent, to be held by said third party until the occurrence of a contingency or performance of a condition, and then to be delivered to the grantee, promisee, or obligee.
Personal data	Any information relating to an identified or identifiable natural person
Data subject	Natural person who can be identified through given personal data.
Author	Writer of a book or article; inventor of something
Rightholder	The person or entity that owns a set of rights (often the copyright) on a given content item.
Patent	The grant of an exclusive right to exploit an invention
Copyright	The exclusive right to reproduce or authorize others to reproduce artistic, dramatic, literary, or musical works
Operation level agreement (OLA)	Agreement on the working relationship between different functional areas within an organization providing IT services to customers
Underpinning contracts (UPC)	Contracts of an IT Service Provider with an external supplier covering the delivery of services that contribute to the delivery of IT services to Customers
Intellectual Property (IP/IP-Rights)	Ownership of something (copyright/ patent/design) which is intangible

5.2 Business/Conceptual

When looking at context from a business perspective, it is possible to adopt several viewpoints. From an actor-centred viewpoint, the context must capture the parameters that make it possible to answer questions that address the concerns of a specific stakeholder. An action context captures information pertaining to the relationship between an actor and an activity. Table 13 depicts some of the parameters that can be derived from adopting this viewpoint.

Table 13: Actor-centric Business/Conceptual parameters

Parameter	Description
Actor	An agent within an organization participating in business processes (i.e. activities)
Resource	From a transformational standpoint, business processes use resources as inputs and outputs. Resources can be tangible or intangible and range from materials to information (examples of resources include the packages involved on a transportation business process, the bar code on a package (as an optical machine-readable representation of data), the actual data that is physically represented by the bar code, etc.). In this case, this parameter represents resources that are inside the boundaries of the organization
Role	The role (s) an actor is playing.
Responsibility	The actor responsible for performing the activity.
Competence	The competences supplied by the actor executing the activity.
Authorization	The authorization given to the actor to execute the activity.
Actor Goal	The goal of the actor performing the activity.
Actor Rule	The business rules that apply to the actor executing the activity.
Actor Location	The location where the actor is placed.

Applying this exercise to digital preservation of business process, from an activity-centred viewpoint and depending on the concerns of the stakeholder in question, the contextual information whose capture might be crucial for preservation may include the parameters expressed in Table 14.

Table 14: Activity-centric Business/Conceptual parameters

Parameter	**Description**
Activity	The activity performed by an actor.
Event	Events related to an activity (start, end, lead time)
Location	The location where the activity is being performed
Goal	The goal (s) the activity aims to reach
Resource	The resources required to perform an activity
Actor	The actors which can execute an activity

From a resource-centred viewpoint, the contextual information includes the context parameters listed in Table 15.

Table 15: Actor-centric Business/Conceptual parameters

Parameter	**Description**
Permission	The permissions associated with a resource
Right	The rights associated with a resource
Actions	The actions that can be performed on a resource
Access	The access options associated with a resource
Restriction	In what action context (see definition above) can this resource be accessed? Example: an actor can be able to access a given resource R when performing some activity A but be restricted to do so while performing activity B.

5.3 System/Logical

Context from a system perspective can be thought of at many levels, from an environmental or user perspective down to a specific context on a local system where software is being executed.

A wider look at context may include the motivations and goals for running a particular piece of software or a collection of software applications from a Business or Enterprise perspective. Software systems will be set up with particular objectives in mind and configured in such a way as to realise these goals. Over time these goals might change and so the use of certain entities within the business process will be made to support these changes. An example would be that as the company grows, there are more users, using the web-page and, therefore, another web-server is required to manage the load. A database may then be added to store user information.

Focusing on the low-level individual machine perspective, the context of how a machine fits the outside world is limited by the software and purpose that it has been assigned (applications running), the current content it is processing and the external interface to the rest of the network. By capturing the software state that is running on the system and the environmental variables we should have a good first level of context. As we build on this, we will need to know the environmental variables, when the applications are run, whether they are deterministic, user inputs, times of running, other concurrent processes and, then, capture the inputs and outputs to the external network. Table 16 describes parameters belonging to the System/Logical perspective.

Table 16: System/Logical context parameters

Parameter	Description
Software Unique identifier	A property uniquely identifying a software (i.e., the NEVRA (name-epoch:version-release.architecture) in Linux systems)
Software	Software applications installed on system (including libraries) or that is available to be installed
Network Connection	Speed, type, bandwidth, limits, etc.
User Authentication	User authentications; whether the user has credentials to run the software at this time
Use of data (Linked in with the user and the process and authentication to be performed)	The use of data in some non-authorised manners can result in a non-authorised running of the software. For example, in some of the Windows licenses, there is a limit to whether an operating system can be run as a Virtual System or not. If this is the fact, it would have to be captured.
License	Is the software that is running, legal in the country in which it is being run? For example, cryptography tools cannot be used outside of certain geographic domains above a certain key-bit level due to trade restrictions.
Authorisation Key	Software sometimes needs to be activated or registered and this is supported by licensing. There may be an issue of expiring certificates, etc.
Timestamp	The current time of an event logged by a computer system

Input/Output parameters	Inputs/outputs to software
Algorithms and heuristics	Algorithms and heuristics used by software; are there dependencies to non-deterministic functions?
Session	Is software run, in a sequence and does timing matter?
Software Input Chain	Input as a chain, captured by an objective for the session or an overall objective
Purpose/Motivation	Purpose/Motivation for running a particular software (i.e., Requirement, Goal, etc.)
Version	Versions of the operating system and programming language / building environment, or of remotely used services
Data	Data sent to and received by software and services, and all intermediate data created by interacting software, to verify the different steps in the process
Data Format	Data formats used (e.g. a content provider might use a different format in the future, with a different encoding, thus losing some information)

Context parameters can also be derived for the specific case of service. As the approached scenarios lacked the emphasis on the service aspect, the SLA@SOI[11] project in which some of the partners participated, implemented a list of Quality of Service terms that can also be considered context parameters for the service aspect. Service-level QoS terms apply to all the virtual machines (VM) which are logically combined and reside within a defined “service”. Table 17 describes service related parameters.

Table 17: Service related context parameters

Parameter	Description
Start Date and Renewal Date	These denote when the service will begin operation and when it will cease. Start Date could be set to a future date or it could be immediate. Renewal Date can be a future point in time which, if reached and the service owner has not renewed the service, it will be automatically taken off line. This can also help avoid issues with VM sprawl.
VM Persistence	This QoS term describes whether the VMs which constitute a service are persistent or not. Persistent virtual machines are stateful, meaning that following a reboot and changes made to the VM since the last reboot will be retained. Non-persistent VMs are state-less and revert back to their original condition if rebooted. This latter feature can be useful for activities such as test spirals, development or training classes.
VM Image	The service owner can choose from standard or encrypted guest templates. Using encrypted templates means that the guest OS within the VM will run disk encryption software. Such protection makes it more difficult for an unauthorised party to boot a VM should they manage to make a copy of its running image or a gain access to a snap-shot or backup on the providers system.
Service Isolation	This QoS term is Boolean and can be used in two ways. Firstly, it can be set to ensure that all VMs belonging to a service are provisioned on separate physical systems from

[11] http://sla-at-soi.eu/

	each other. It can, at the providers discretion, also be used to instead ensure that a service is given dedicated hosts which are not shared with VMs from other services which can be a useful security feature helping to alleviate concerns around co-tenancy.
Auditability	This QoS term is another Boolean value and is used if the customer required that a service audit trail is required. A service audit trail simply means that all administrative operations relating to that service are stored in a security log and made available to the customer.
SAS70 Compliance	This is a guarantee term designed for enterprises that need to consider compliance with Statements on Auditing Standards (SAS) number 70. It is used to control where within the cloud the service is allowed to run. Internal compliance procedures must be carried out by the provider on the physical hosts. The provider may carry these out on all systems in the cloud, or for Total Cost of Ownership (TCO) reasons, may just do this on the most sensitive ones. This term therefore allows the scheduler the flexibility to run services on systems which are not SAS70 compliant, should data privacy not be as important to the customer.
Service Data Classification	Data stored within the VM can be classified as either "Secret", "Confidential" or "Public" or by some other provider defined terms. This allows the provider to make decisions about where that data is physically allowed to reside in order to comply with the corporation’s internal data protection policies.
Service Data Retention	This QoS term, specified by the service owner, relates to the number of days which the provider will agree to maintain the customer’s data after the service ceases to be actively running. If set to 0, the data is deleted from the providers repositories as soon as the SLA contract ceases and the customer has elected not to extend the end date. If set to a value such as 2555 (i.e.: 7 years) the provider agrees to store the data for this period of time. The customer may then be able to comply with legal requirements they may be subject to on data retention.
Service Data Delete Method	This term defines HOW data is erased from the provider’s disk once the service owner no longer requires it to be stored. The provider defines what each level means. Possible values may be “Standard” or “Secure”. Standard may mean a normal delete which does not overwrite the areas of disk where the data used to reside with random 1’s and 0’s. There is an assumption that there is a compute resource cost to using the secure delete method from the provider. However, the provider may elect to use secure delete in all cases and would therefore not offer a choice, but rather advertise it as a security feature. This feature makes it more difficult for third parties to gain unauthorised access to data.
VM Snapshot Backup and VM Snapshot Retention	These QoS terms allow the service owner to request that regular snapshots of their data be taken. If this is required, the retention term lets the service owner specify how long they require the provider to keep each snapshot available to them. This lets the service owner be prudent about older images and ensure they are not stored for longer than required, which makes it a little easier to protect against unauthorised access.
Service Availability Restrictions	Recurring time windows during which service violations will not be considered to also violate the SLA agreement. These windows are defined by the customer and used by the provider to perform maintenance with notice given in advance. Possible values=24x7, 8x5, weekends only

Service Hardware Redundancy Level	Provider defines what is meant be each level in the context of the available hardware within their cloud. Possible values are defined by the provider and can be generic (e.g.: Standard, Better, Best) or specific (full HW redundancy with a standby node, some HW redundancy (RAID, memory mirroring etc.) or no HW redundancy). Each of these solutions can incur additional costs for the provider.
Controlled Country Regions	Permitted if this VM is allowed to run in a technology Controlled Country Regions (CCRs). CCRs are defined due to political reasons and multinationals operating in CCR geographies must ensure that intellectual property-leaks to native industry are prevented in these countries.
Data Encryption	Enabled if the provider should encrypt the VM HD files at the guest OS level (using, for example, PGP) so they can't be booted by a 3rd party. Note: this means that the VM HDD files are encrypted so that if a 3rd party obtains the VM file, they cannot boot the VM without knowing the pass phrase or certificate key.

5.4 Technology/Physical

As for the technology perspective on context, Table 18 provides examples of possible parameters, in its majority derived from the SLA@SOI project, as the scenarios lacked focus on the physical/hardware aspects of context.

Table 18: Technology/Physical context parameters

Parameter	Description
Hardware	Hardware that the software is running on, CPU (speed, architecture, etc.) /memory (speed, type, size etc.), Graphics Card, Network Interface, etc.
Location	Provider defined or automatically detected, this attribute identifies the location of the physical host server. It can be a geographic region, a country, a site, a building within a site, or a data centre within a building.
CCR	Specifies whether the physical host resides in a CCR region.
Efficiency	This is a rating of compute power versus total cost of ownership (TCO). Higher ratings mean that the server provides more capacity per euro spent supporting it. If run-time decisions are made to consolidate workloads, the least efficient servers are the first to be powered off. Ratings are set by the provider. SLA@SOI showed how these can be calculated
hwRedund	Specifies the level of redundancy which the host server supports and is matched to the levels in the service offerings.
DiskThruput	Specifies the level of disk throughput which the host server supports and is matched to the levels in the service offerings.
NetThrouput	Specifies the level of network throughput which the host server supports and is matched to the levels in the service offerings
SAS70	Specifies whether this host sever is covered by SAS70 (i.e.: IT server administrators carry out SAS70 audits of this physical server periodically).
DataClass	Specifies the level of data classification that the host server is cleared to support and is matched to the levels in the service offerings. In the background, IT admins can set up firewall rules between the servers and set local access rights to restrict access in accordance with the dissemination level that is assigned to the server.
Intel(R) TXT(R)	If an additional layer of security is requested, this term can be used to ensure that a service is only ever allowed to run on infrastructure which supports Intel's Trusted Execution Technology (TXT) which ensures that the physical host BIOS, operating system or hyper-visor have not been compromised or tampered with. Automatically detected or manually set, this context attribute specifies whether the host supports this technology in its chipset.
Intel(R) AEN-NI(R)	Advanced Encryption Standard New Instruction (AES-NI) is an instruction set which has been added to Intel Xeon processors to speed up the encryption of data at rest, in an application or in transit. Often today's security applications implement a software AES solution which is slower. Using this QoS term can ensure that, if any encryption software running within the VMs uses AES-NI, the hardware will support it and thus

	speed up encryption tasks. Automatically detected or manually set, this context attribute specifies whether the host supports this technology in its chipset.

6 Context Model

As illustrated in the related work section 3.2.1, ontologies are formal, explicit specifications of shared conceptualizations for a domain of interest (Staab and Studer, 2009) and they provide the ability to formally specify, classes, individuals and their relationships. Ontologies meet the TIMBUS requirement of providing a modelling approach that is able to capture instances of context parameters (which is the focus of deliverable D4.5) and their inter-relationships (which is the focus of deliverable D4.2). In our ontology implementation, a context parameter is implemented as a class conceptualization (in ontology terminology), a relationship between context parameters is implemented as an object property concept (in OWL terminology), and particular instances of a context parameter (e.g. a specific server which is the instance of the server context parameter) are implemented as individuals (in OWL terminology).

In consequence, the “Context Model“ actually defines a meta-model which can be instantiated to specific models. In the context of TIMBUS, this meta model defines what our shared conceptualization of business processes and their context (with focus on their digital preservation) looks like. Therefore, it provides the conceptual framework which defines how valid “Context Model Instances“ of the meta-model (applied on specific business processes) look like.

In future discussions, we therefore define the following terminology which is mandatory for the TIMBUS project:

- “Context Model“: This term refers to the meta-model. It is characterized by containing the knowledge (of all TIMBUS partners) which defines what a business process in the relevant context of its digital preservation looks like. The model does not contain any classes, relations or individuals which are specific to a particular business process.
- “Context Parameter“: Each class in the Context Model, which represents a relevant parameter of the context of business processes, is called a “Context Parameter“.
- “Context Model Instance“: This term refers to an instantiation of the meta-model. It is characterized by its application to a particular business process. The model instance defines the relevant (from a digital preservation perspective) entities and dependencies which define the context of a particular business process in focus. In other words, a context model instance represents a knowledge base of relevant (from a digital preservation perspective) contextual knowledge of a particular business process.
- “Context Parameter Instance“: An instance of a Context Parameter is called a Context Parameter Instance. In essence, a Context Parameter Instance, in ontology terms, is an individual in a Context Model Instance ontology. Context Parameter Instances can therefore be only elements of Context Model Instances.

Furthermore, ontologies are an ideal candidate for context modelling in TIMBUS, as reasoning functionalities (in form of queries on knowledge bases) are an integral part of TIMBUS, such as:

- Is a business process preservable?
  - What parts are entirely automatically preservable?
  - What parts are semi-automatically preservable?
  - What parts are only manually preservable?
- Can we move a business processes from our country to another country?
- Does a new license conflict with licenses already relevant to a process?
- ...

As was introduced earlier, ontologies provide a model-theory grounded semantics which enables complete and sound reasoning on ontologies , by, for example, answering queries to a knowledge base as in the above list. We envision addressing reasoning problems relevant to TIMBUS using generic reasoning mechanisms which are, for example, available in state-of-the-art ontology formalisms (e.g. OWL v2.0).

6.1 Design Methodology

The Context Model has been designed to provide a holistic view on the relevant context parameters of a business process from a digital preservation perspective. As the context model is actually a meta-model and has to be instantiated for specific business processes, the meta-model, in theory, basically would have to contain the entire domain knowledge required to model all possibly imaginable business processes from the digital preservation perspective relevant to TIMBUS. This is quite similar to "modelling the entire world" which literally seems to be never-ending task, as seen in other ontology modelling projects.

Therefore, to streamline our design process of the meta-model, the knowledge on digital preservation relevant context parameters has been captured in a problem-motivated approach (as also motivated in the introduction section 2.3) based on the investigation on TIMBUS partners' relevant scenarios/use cases. This knowledge has afterwards been transformed into an OWL ontology, called the Context Model. The Context Model (meta-model) can be instantiated into Context Model Instances (instance models) to apply the Context Model to concrete business processes during the process of their digital preservation or the planning of their digital preservation.

This establishes an iterative approach of refinement of the Context Model, based on relevant scenario investigation, application (i.e. instantiation) of the Context Model for these scenarios, and refinement of the meta-model afterwards, in order to incorporate the lessons learned.

6.1.1 Terminology Conventions

To streamline our approach and discussions on modelling Context Models and Context Model Instances, we choose the following modelling terminology for the TIMBUS project:

- "Domain of interest": The domain on which we model the knowledge captured in the ontology. In OWL terminology this would be the "Domain of Discourse".
- "Ontology", "Terminological Knowledge" and "Asserted Knowledge": An ontology is a model of our domain of interest. An ontology consists of instances, classes and relations. As illustrated above, we have a Context Model and Context Model Instances, all of which are ontologies. They differ in the knowledge they capture. As mentioned, the Context Model is a meta-model, it basically captures the "terminological knowledge" of our domain of interest. This terminological knowledge provides the ability for us to talk about our domain of interest in general. Furthermore, the Context Model can be instantiated during the application to a business process. This essentially means that the context model is populated by concrete instances which are part of our domain of interest. This additionally captured knowledge (by our Context Model Instances, or by ontologies that "talk about concrete individuals" in general) is called "asserted knowledge". In OWL terminology these would be an "Ontology", the "T-Box" and the "A-Box".
- "Instance": A particular individual/instance in our world/domain of interest. For example, the one and only computer on my desk. In OWL terminology this would be an "Individual".
- "Class" and "Sub-Class": A class is a set of instances which have common characteristics. For example, all built computers (i.e. all instances which are a computer) have, for example, in common that they are computers (which is true in our domain of interest). Therefore, all computers are instances of the "Computer" class. In other words: they are of type "Computer". Equally true is that all these computers are instances of the "Thing" class. But, as there are more things than computers in our domain of interest, the "Computer" class is a sub-class of "Thing". In OWL terminology these would be "entities".
- "Relation": Relations describe relationships between classes and individuals. When applied to classes, relations define the common characteristics shared by all individuals which are instances of these classes. When applied to individuals, relations define the common characteristics shared by specific individuals. In essence, relations define the characteristics that all individuals of a class (or type) have in common
- "Dependencies": In TIMBUS, there is a special set of relations referred to as "dependencies". These relations describe generic dependency relations between classes and individuals in our domain of interest. For example, the relations "depends" and "conflicts" are part of this set. What these dependencies informally and formally mean to specific classes or individuals has to be declared in the ontology in the context of these specific classes or individuals. This can be done informally through a textual and human-readable description and formally using rules or axioms (which are introduced in related works section 3.2.1).
    - "depends": Declares that arbitrary classes or individuals depend on each other.
    - "conflicts": Declares that arbitrary classes or individuals conflict with each other.

6.1.2 Design Conventions

Constructing "well-structured" and "well-named" hierarchies of classes and relations, and classifications of instances, is a complicated process. Ontology designers (also called knowledge engineers) have a subjective perspective in modelling the knowledge on a part of the world (their domain of interest). Their perspectives may differ on what are the relevant parts of the domain of interest, and in their perception on the suitable granularity of classes and relations in the hierarchy. This can easily lead to inconsistencies where classes, instances and relations should be placed in the hierarchies "to best model" the domain of interest. To streamline the Context Model and Context Model Instances modelling in TIMBUS, in the following, conventions for modelling the Context Model and Context Model Instances are established.

Firstly, naming conventions specify the proper naming of classes, instances, relations and dependencies in TIMBUS. And secondly, design pattern conventions establish best practices in modelling frequent problems, for example, "how to represent information objects, such as a text, and its information representations, i.e. its physical representations, such as a PDF or a printed book" and "how to establish a new class in the hierarchy".

6.1.2.1 Naming Conventions

With ontology formalisms, as for example OWL, usually each ontology consists of only one global name space. Therefore, to uniquely identify an element of an ontology, the name of each element has to be unique in this ontology. Therefore, each class name, individual name and relation name has to be unique in the Context Model in TIMBUS. To prevent TIMBUS partners from creating naming conflicts the following guidelines have been established for naming classes, individuals and relations:

6.1.2.1.1 Classes

Each class name is prefixed by "c_", indicating that this name belongs to a class relation, and to disambiguate it from individuals and relations. For example: "c_Fruit" refers to the class of all fruits.

Each class name must be defined by a precise term, as, for example, "Fruit" to refer to the class of fruits. "Fruits" would be a term consisting of a single word. In TIMBUS, each word contained in a term starts with a upper-case first letter and continues with lower-case letters. For terms consisting of a sequence of words the camel-caps notation is to be applied, for example: "SportsEquipment".

In TIMBUS, there must never be a naming conflict between direct sub-classes of a particular class. But, as the English language contains homographs, there may exist naming-conflicts between classes which do not directly inherit from the same super-class. For example, "bat" may be the class of all bat-style sports equipments, or the class of all bat-style animals. In these cases, in TIMBUS, the minimum required classes' contexts in the class hierarchy are incorporated as prefixes to differentiate their names. Classes context refers to their chain of super-classes.

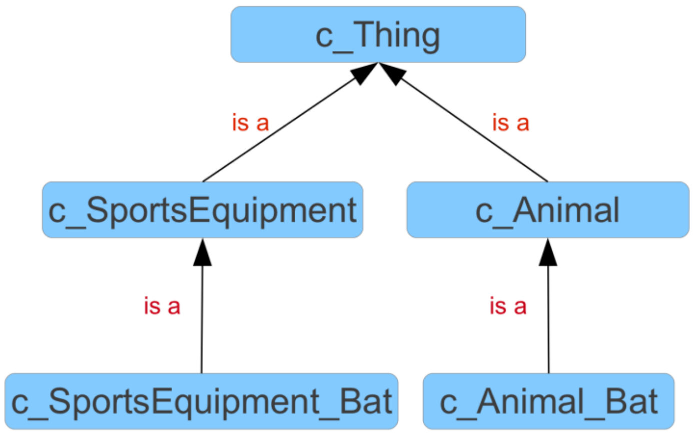


Figure 28: Disambiguating Bat Class Names Example

For example: in Figure 28, “c_SportsEquipment_Bat” refers to the class of all sport bats, and “c_Animal_Bat” refers to the class of all bat animals. These two class names are disambiguated by incorporating the name of their parent classes. As their parent classes are two different ones, this always suffices to disambiguate the two class names.

Syntactically, the prefixes between “c_” and the class name are constructed by concatenating the super class name to “c_” and inserting another “_” between the super class name and the class, for example, “c_SportsEquipment_Bat”.

6.1.2.1.2 Individuals

Each individual name is prefixed by "i_", indicating that this name belongs to an individual relation, and to disambiguate it from classes and relations.

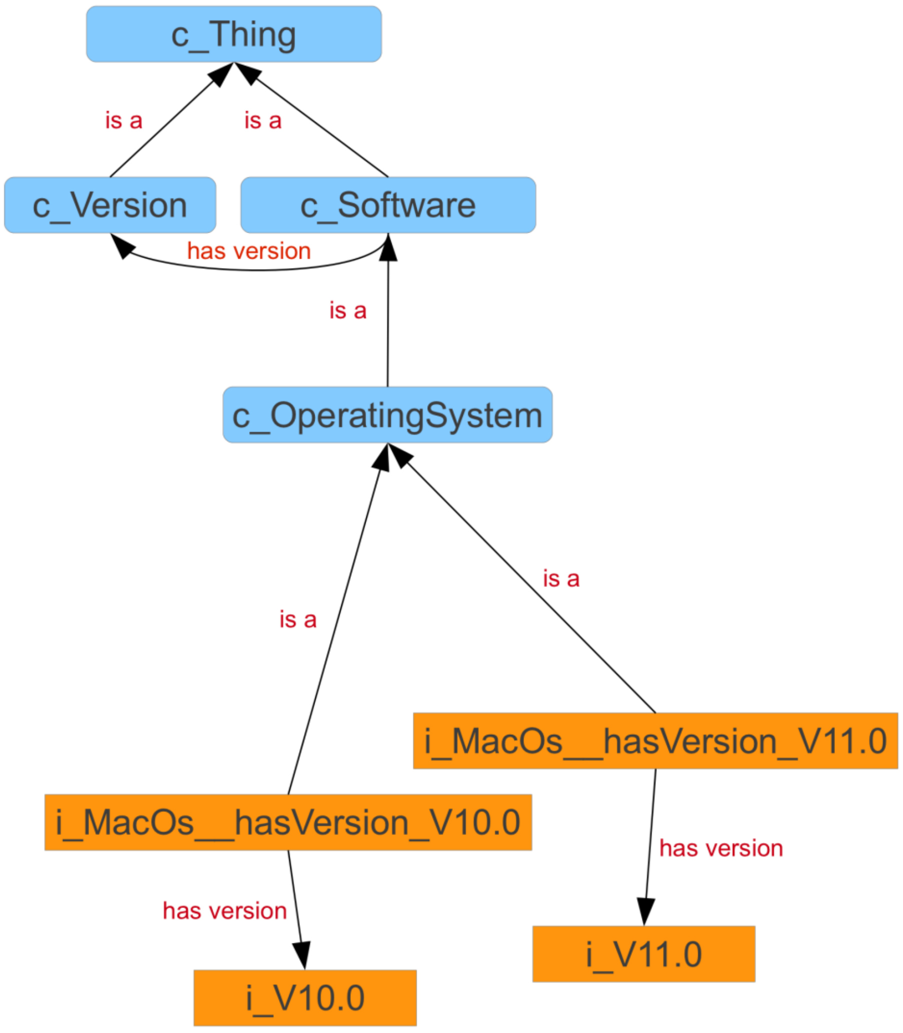


Figure 29: Disambiguating Instance Names of the Same Direct Super-Class

Each individual name mustbe defined by a precise term, as, for example, "JohnSmith" to refer to the one and only John Smith in our domain of interest. As with classes, in TIMBUS, each word contained in a term starts with an upper-case first letter and continues with lower-case letters. For terms consisting of a sequence of words, the camel-caps notation is to be applied, for example: "JohnSmith".

In contrast to class names, it is easily possible that two distinct individuals of the same class share the same name. For example, the individual name "MacOs" might refer to distinct instances of the Mac Operating System. In TIMBUS, name-conflicts between individuals of the same class are resolved using suffixes that reflect the relations of these individuals. The relations of the individuals are incorporated into disambiguation in alphabetical order. Furthermore, only the minimal set of relations required to disambiguate all individuals in a class must be incorporated.

For example, in Figure 29, “i_MacOs__hasVersion_V10.0” refers to the individual of Mac OS in version 10, and “i_MacOs__hasVersion_V11.0” refers to the individual of Mac OS in version 11. These two individual names are disambiguated incorporating only their “hasVersion” relation.

In the case of our “JohnSmith” example, we might have different JohnSmiths in different geographic locations. They could, for example, be differented in our domain of interest using these individual names: “JohnSmith__livesIn_London” and “JohnSmith__livesIn_NewYork”.

It is possible, that the available relations of individuals do not suffice to syntactically disambiguate all individuals in a class. In this case, it is necessary to add further relations to the ontology.

Syntactically, the suffix for each relation after an individual's name is constructed by concatenating two underscores (“__”), the relation's name (e.g. “hasVersion”), one underscores (“_”) and the related individual's name (e.g. “V10.0”).

A syntactic sequence of relations, to disambiguate two individuals, is constructed by iteratively applying the previous rule. Relations have to be applied in alphabetical order, even if the application of a relation does not disambiguate the individuals.

As with classes, it is possible that any two individuals of different classes conflict in their names. This seems unlikely after the class-internal name conflicts between individuals have been resolved, but it is still possible. In this case, the same rules to disambiguate class names is to be applied to names of individuals, i.e. instance names are to be prefixed by the minimal set of class hierarchy context that suffices to disambiguate any names of instances.

6.1.2.1.3 Relations

Each relation name is prefixed by “r_”, indicating that this name belongs to a relation, and to disambiguate it from classes and individuals.

Each relation must be defined by a precise term, as, for example, “hasVersion” to refer to the relationship between things and their versions. In contrast to classes and individuals, in TIMBUS, the first word contained in a term starts with a lower-case first letter and continues with lower-case letters. But all other words in a term, as with class and individual names, start with a upper-case first letter and continue with lower-case letters. For terms consisting of a sequence of words the camel-caps notation is to be applied from the second word on, for example: “hasVersion”.

Furthermore, relation names should be chosen in a particular pattern. As shown in the following section , there should always be a relation and its inverse relation to foster reasoning. To enable intuitive association of a relation and its inverse based on their names, a relation should always be named in a way so that its inverse is easily syntactically recognizable. The following naming patterns are mandatory to TIMBUS:

Table 19: Patterns for Relation Naming

Relation Name Pattern	Inverse Relation Name Pattern
"has..." (e.g. hasVersion)	"is...Of" (e.g. isVersionOf)
"..." (e.g. RefersTo, knows)	"is...By" (e.g. IsReferredToBy, isKnownBy)

As with classes, in TIMBUS, there must never be a naming conflict between direct sub-relations of a particular relation. But as the English language contains homographs, there may exist naming-conflicts between relations which do not directly inherit from the same super-relation. For example, "isAfraidOfBats" may be a relation between people and sport bats, or the relation between people and bat animals. As with classes, in these cases, in TIMBUS, the minimum required relations' contexts in the relation hierarchy are incorporated as prefixes to differentiate their names. Relation context refers to their chain of super-relations.

Syntactically, the prefixes between "r_" and the relation name are constructed by concatenating the super relationname to "r_" and inserting another "_" between the super relation name and the class, for example, "r_hasIngredient_hasSpice".

6.1.2.2 Design Pattern Conventions

The following design patterns are to be applied to the Context Model and Context Model Instances whenever possible. They provide guidance to frequently occurring design problems in our TIMBUS ontologies.

6.1.2.2.1 Informal Descriptions

All classes, individuals and properties in the Context Model and Context Model Instances need to be associated by two descriptions, a brief descriptions and a long description.

Both descriptions should provide an intuition on the meaning of the elements. The brief one provides an abstract description in a single sentence, and the long one provides a more elaborate one using several more concrete sentences. These descriptions are to be added using associations in OWL. These provide the ability to associate additional information to the elements in an OWL ontology. For this purpose, two OWL associations, called "briefDescription" and "longDescription" are to be used, both of which associate an English text with ontology elements, i.e. classes, individuals and relations.

6.1.2.2.2 Uniform Resource Identifiers

Whenever, in the TIMBUS Context Model, an individual is to be related to a Uniform Resource Identifier (URI), the relation "hasUri" is to be used which associates this individual to another individual whose name is the URI. In OWL, this has to be designed by a "data property" called "hasUri" which associates "Things" to the String data type.

6.1.2.2.3 Information Realization Pattern

In this section the information realization design pattern is illustrated. The pattern addresses "relations between information objects like poems, songs, formulas, etc., and their physical realizations like printed books, registered tracks, physical files, etc.." (Presutti and Gangemi, 2008)

Figure 30 depicts the issue of modelling operating systems and their installations. In this example, the operating system has the role of an information object and installations (on some platforms) are their physical realization. In the figure you see two different releases of Mac OS, V10.0 and V11.0, both of which are named according to the conventions illustrated earlier in section 6.1.2.1. In particular, in this ontology there are not many instances of Mac OS V11.0, one for each of its installations, but only one, which is called "i_MacOs__hasVersion_V11.0". The name contains the "hasVersion" relation to differentiate it from the Mac OS V10.0 instance. For each installation, i.e. physical realization of "i_MacOs__hasVersion_V11.0", there exists an individual of type "InformationRealization". In this example, there exists only one physical realization: the Mac OS V11.0 installation on the notebook of Alex. (This notebook is abstractly modelled as a platform.)

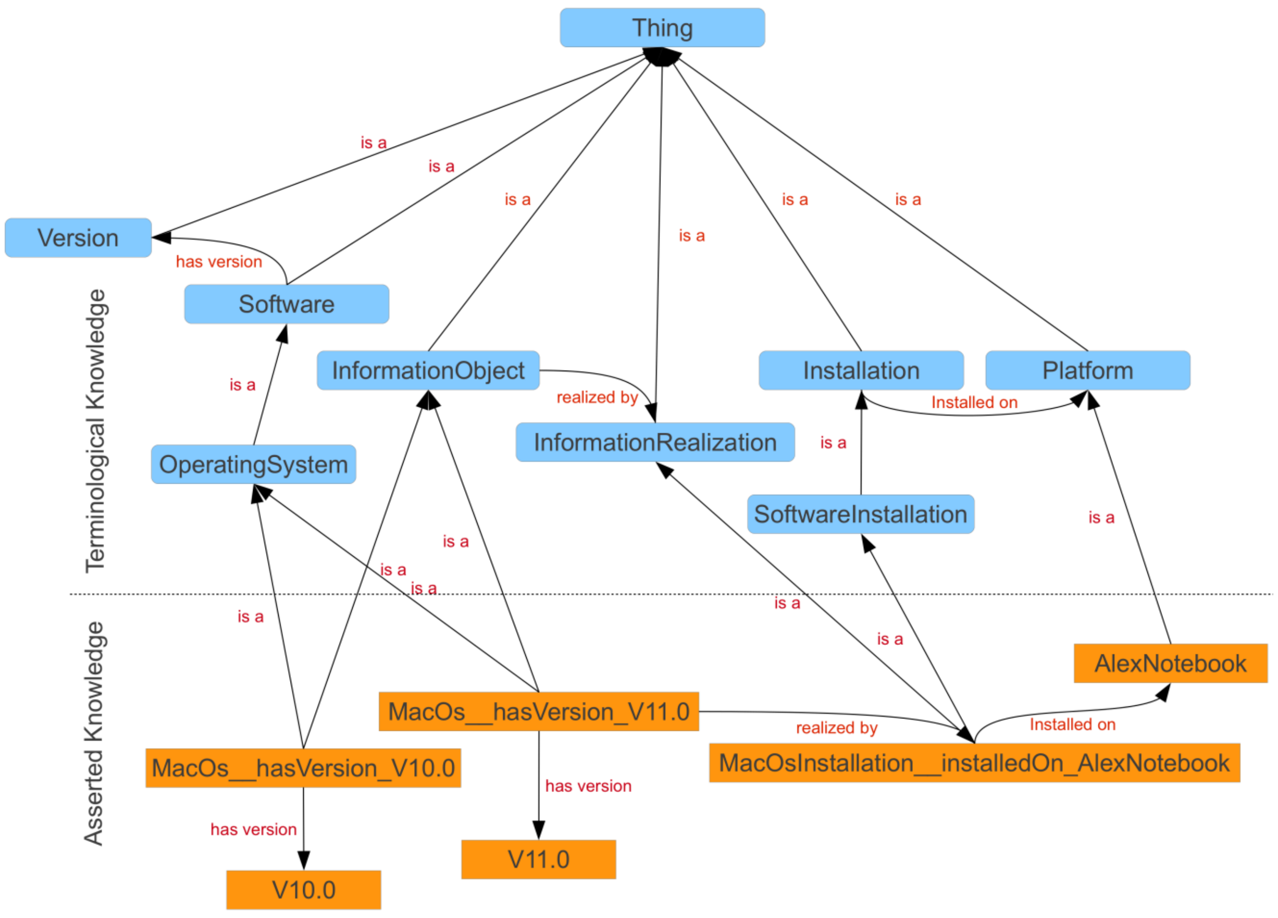


Figure 30: Information Realization Pattern

6.1.3 Refinement Process

The Context Model and Context Model Instances are to be kept in the project's content versioning system and are to be continuously refined to be coherent with the scenarios in deliverable D4.5 and D4.9. But as (to the best of our knowledge) merging OWL ontologies purely on the basis of any available textual representation is a quite complex manual task, TIMBUS partners may not work in parallel on the OWL model, but instead in an interleaving fashion. To achieve this without high communication overhead between parties, the OWL models in the project's content versioning system have to be locked for exclusive write-access by the partner who wants to contribute to the models.

6.1.4 Versioning und Publication Process

The Context Model and the Context Model Instances are to be named, versioned and published (project internally) in a consistent manner. The following example serves as a template:

http://timbus.teco.edu/ontologies/2011/11/ContextModel.owl

The Context Model is to be published on the project website, using above URL as a template. In this example, the URL refers to the Context Model published at some time in November 2011. This implicitly declares its version, the Context Model in Version: November 2011. Each month, there may be only one publication of a model, to prevent naming-conflicts.

In addition, each model requires a brief and specific names, as, for example, “ContextModel” for the Context Model. In case the name consists of multiple words, the camel-caps notation is to be applied.

6.1.5 Representation and Visualization

The Context Model and Context Model Instances are to be syntactically represented and visualized in a consistent way.

- Files: In case the ontologies are to be stored, for example, for exchange between parties or systems, the OWL/XML format standard is to be used.
- Documents: In case the ontologies, or excerpts of the ontologies, are to be visualized or represented in documents, a graph-based visualization is to be provided. The employed tools are not prescribed. Nevertheless, the “OntoGraf”[12] tool is recommended. OntoGraf is integrated into the Protege[13] ontology modelling suite.

[12] http://protegewiki.stanford.edu/wiki/OntoGraf

[13] http://protegewiki.stanford.edu/wiki

6.2 Initial Version

As mentioned in the goals of the deliverable in Section 2.2, a challenge faced in TIMBUS is the large domain of parameters that may be potentially relevant to the context of a process. On the one hand, the problem domain includes a potentially infinite set of relevant parameters at the same level of granularity. And on the other hand, the potential levels of granularity seem to be infinite too. Therefore, to scope and to structure the exploration of relevant context parameters, the investigation has used a divide-and-conquer approach: The Top-Down approach of using Zachman as top-level ontology, and the Bottom-Up approach of using scenarios for detailed ontology meet in 2nd top level ontology. This approach is akin to that suggested of a middle-out approach taken by Uschold and Gruningers' methodology (Uschold M., et al., 2006).

The current release of the Context Model can be found at:

http://timbus.teco.edu/ontologies/2012/04/ContextModel.owl

6.2.1 1st Top-Level Ontology: Zachman

To guide the top-down perspective, related work on established enterprise frameworks has been analysed. These models provide a holistic but abstract view on the relevant concerns of an enterprise – views which incorporate business processes and relevant aspects from an enterprise perspective. From an ontology perspective, the enterprise framework serves as a top-level ontology. Our analysis has selected the Zachman framework, as shown in Figure 31, to be most suitable, as it covers business processes and related relevant aspects from various different perspectives which focus on different but distinct concerns. This way our top-down efforts can be thematically partitioned.

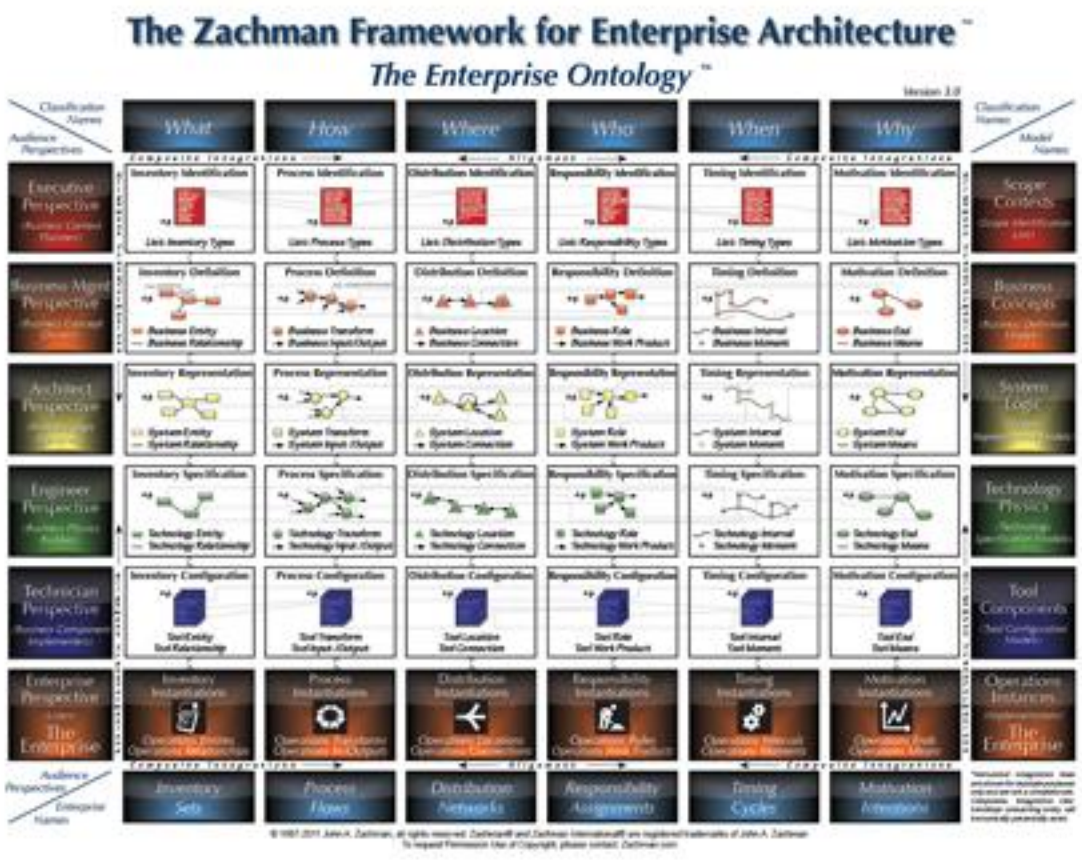


Figure 31: Zachman Framework v3.0

Source: http://www.zachman.com

The current release of the Zachman Framework ontology facilitated in TIMBUS can be found at:

http://timbus.teco.edu/ontologies/2012/04/Enterprise.owl

The model tries to capture the Zachman ontology in a holistic way to provide the ability to look at relevant context parameters from various perspectives. To enable this, the context parameters modelled in section 6.2.3 are associated with the cells that comprise the Zachman framework. This way, when looking from a particular Zachman perspective (e.g. from a business planner's perspective), the relevant context parameters can be determined. As introduced in section 3.1.1, the following perspectives established in Zachman have been modelled:

1 Enterprise Audience Perspective (Represents the concept of audience perspectives (6 perspectives) onto an enterprise (called “Audience Perspectives” in the Zachman Enterprise Framework 3.0).)

   1.1 ExecutivePerspective

   1.2 BusinessManagementPerspective

   1.3 ArchitectPerspective

   1.4 EngineerPerspective

   1.5 TechnicianPerspective

   1.6 EnterprisePerspective

2 Enterprise Dimension (Represents the concept of a 5W1H perspective (6 dimensions) onto an enterprise (called ``Classification Names'' in the Zachman Enterprise Framework 3.0).)

   2.1 WhatDimension

   2.2 HowDimension

   2.3 WhereDimension

   2.4 WhoDimension

   2.5 WhenDimension

   2.6 WhyDimension

3 EnterpriseLayer (Represents the concept of a horizontally layered perspective (6 layers) onto an enterprise (called ``Model Names'' in the Zachman Enterprise Framework 3.0).)

   3.1 ScopeContextsLayer

   3.2 BusinessConceptsLayer

   3.3 SystemLogicLayer

   3.4 TechnologyPhysicsLayer

   3.5 ToolComponentsLayer

   3.6 EnterpriseLayer

Each of these perspectives provides a scoping to relevant context parameters and a partitioning on the context parameters in the Context Model. The scoping restricts our efforts to the aspects relevant to Zachman and the partitioning, using the relationships of context parameters to cells in the Zachman framework, ensures that we identify “over-heated” (many parameters in one cell) and “cold” (few or no parameters associated to a cell) spots in our model. Hot spots are interpreted as indicators for over specification of context parameters, whereas cold spots are used as indicators for under specification.

6.2.2 2nd Top-Level Ontology: Where Context Parameters and Zachman Meet

The ontology shown in Figure 32 represents the integration of our bottom-up efforts on modelling relevant context parameters (driven by related work on modelling and digital preservation) and of our top-down efforts to interpret the framework established by Zachman.

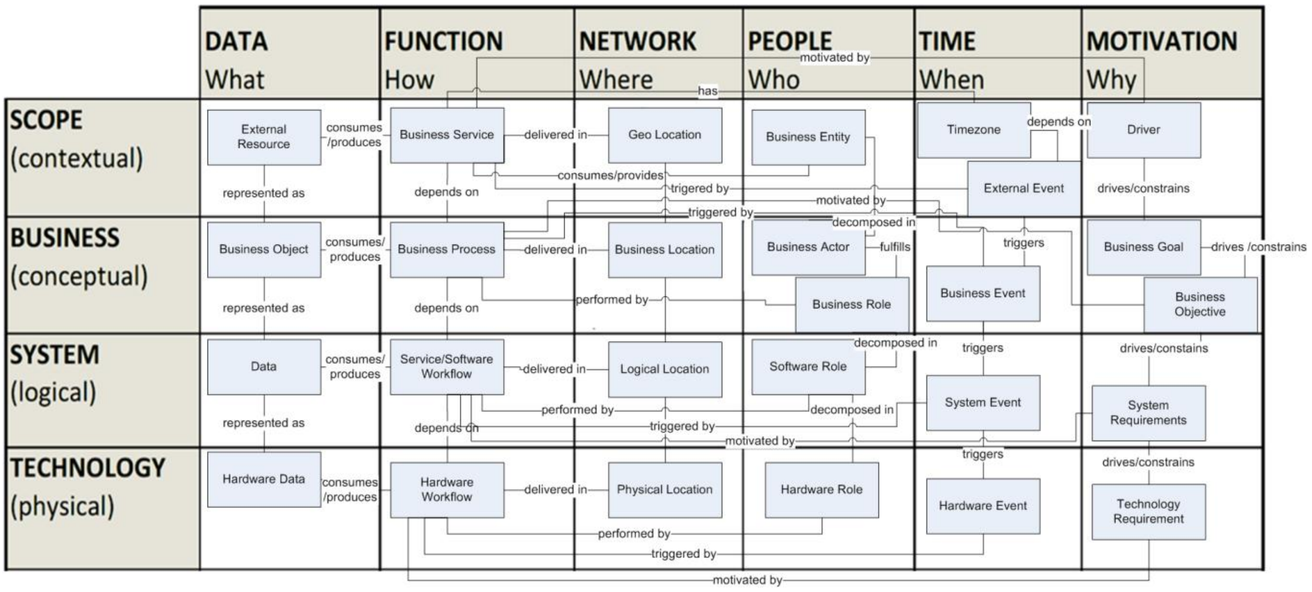


Figure 32: 2nd Top-Level Ontology in the Context Model

6.2.3 Context Parameter Ontology

To guide the bottom-up perspective, related work on modelling approaches that are related to TIMBUS, as for example the PREMIS data dictionary, and other works that serve as a source for digital preservation-relevant aspects have been investigated. With these backgrounds in mind, partner-specific scenarios for business process preservation have been designed and their context parameters have been identified. Parts of the Context Model ontology are now presented, focusing on excerpts that reflect the four categories of context parameters presented in the previous section 5.

6.2.3.1 Scope/Context

Legal relations in an Enterprise are fairly difficult to represent in their entirety. From previous research into the state of the art there is no single ontology that represents all legal information. However in Figure 33, certain contextual parameters have been identified and some relations between them proposed. In terms of Digital Preservation as the motivating question, certain assumptions can be made and more generic relations stipulated. This includes classifying Service Level Agreements and Escrow Agreements as types of Contracts which right-holders and certain legal entities (data subjects) have to comply to. Other terms such as copyright are similar to licences and contracts but have a different area of focus and for the question of whether a process is preservable have to be answered separately.

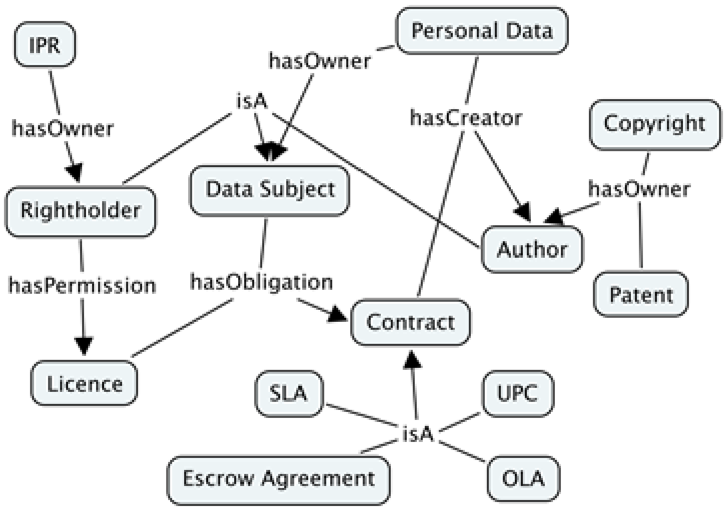


Figure 33: Context Model Part on Legal Context Parameters

6.2.3.2 Business/Conceptual

An Enterprise can have a resource centric perspective where resources are only accessible after certain conditions have been met. The access to resources can be seen in Figure 34. Only by having certain permissions and access or rights can a resource be used and even then there may be certain limitations imposed on how the resources are used.

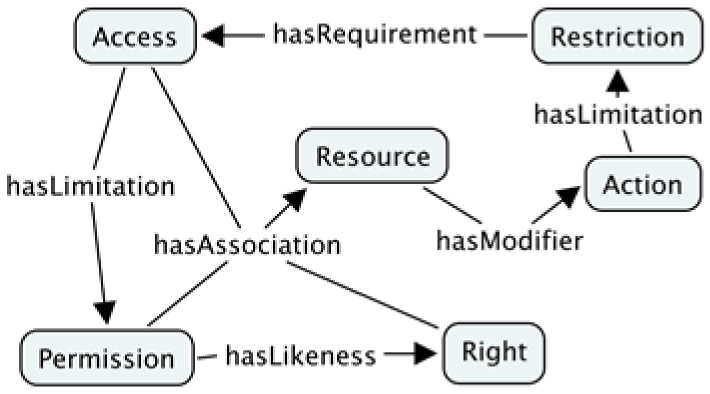


Figure 34: Context Model Part on Resources

In an actor-centric perspective of the business, the relations all centre around actors as can be seen in Figure 35. Actors have a set of attributes that can be recorded as relations to various context parameters that are more generic such as role and responsibility. The relations for an actor help to better specify what the specific actor is doing. By further describing what an actor is, it helps to limit the potential domains in which an actor is relevant and become more specific about its behaviour.

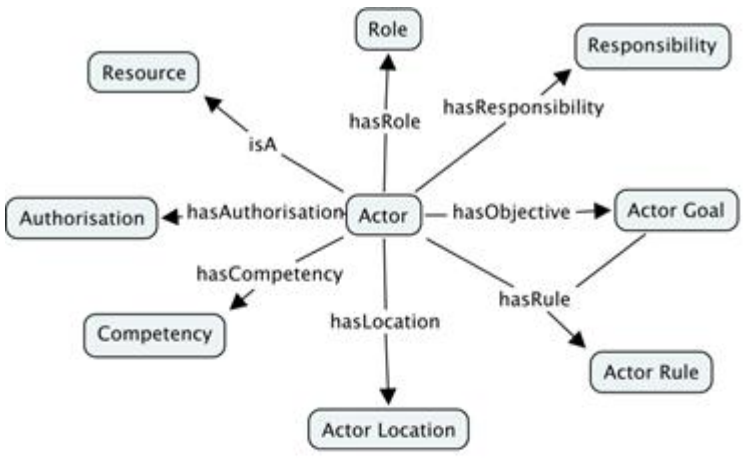


Figure 35: Context Model Part on Actors

Software and People (or actors) are shown in Figure 36 as being the agents for executing an activity. An activity has a goal and can be recorded in terms of event milestones. The executors of an activity can be viewed as special types of resource in an Enterprise and in the example it is expected that resources are modified as a result of the activity. The executors of an activity are complements that work towards a goal. Software and actors are differentiated by the fact that software has been developed or created by someone and is usually specific to one type of activity.

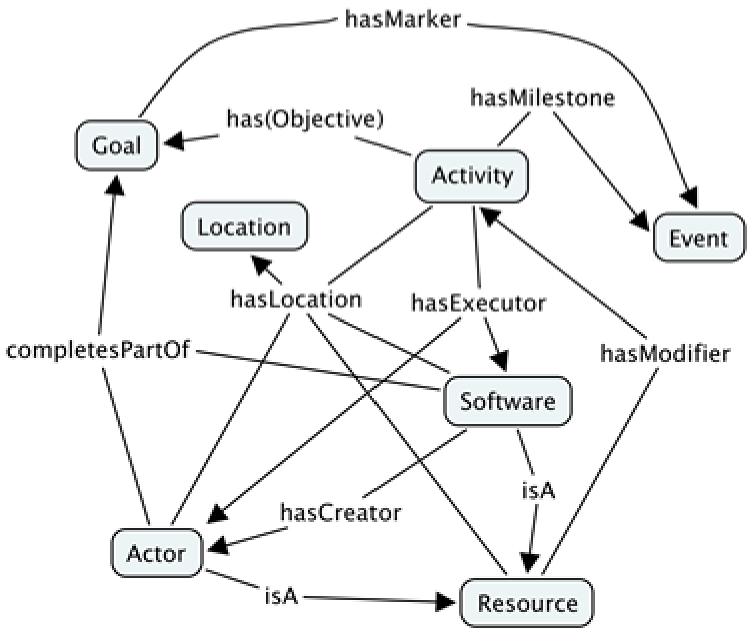


Figure 36: Context Model Part on Activities

6.2.3.3 System/Logical

Quality of Service terms can be applied to services as was implemented by the SLA@SOI project. In Figure 37 we can see how some of these contextual parameters are related. For instance Virtual Machines that are on a server can be encrypted using an encryption mechanism in accordance with best practice or in terms of Governance/Regulation or Compliance frameworks such as the SAS70 Compliance framework. Other important relations to capture for a Virtual Machine are the state of the Virtual Machine and being able to take a snapshot for purposes of migration, provisioning or backing up. Other considerations for the QoS parameters are regular backups for the purpose of audits that have to be performed at certain times for the process to be valid.

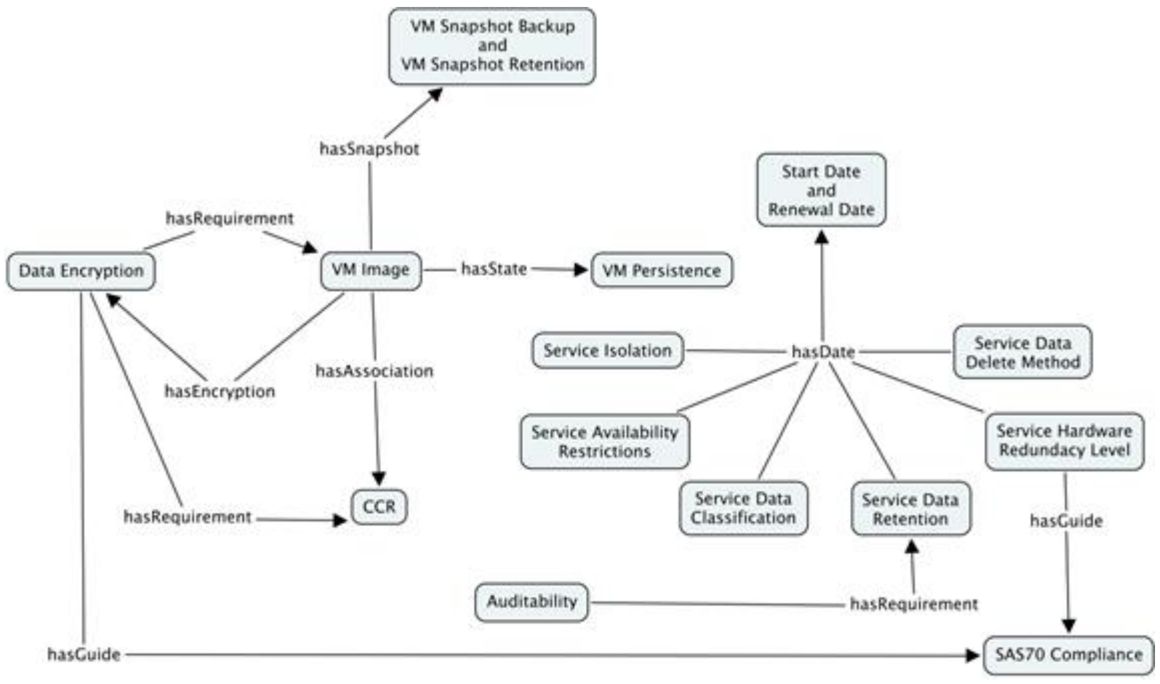


Figure 37: Context Model Part on Virtual Machines

6.2.3.4 Technology/Physical

For the technology perspectives, SLA@SOI has additional parameters identified. As shown in Figure 38, hardware has some measurable parameters such as Disk and Network throughput that can be recorded as metrics from which Digital Preservation can be verified against the expected behaviour. Certain frameworks can be used as guidance and for the detection of hardware including SAS70 and Intel TXT.

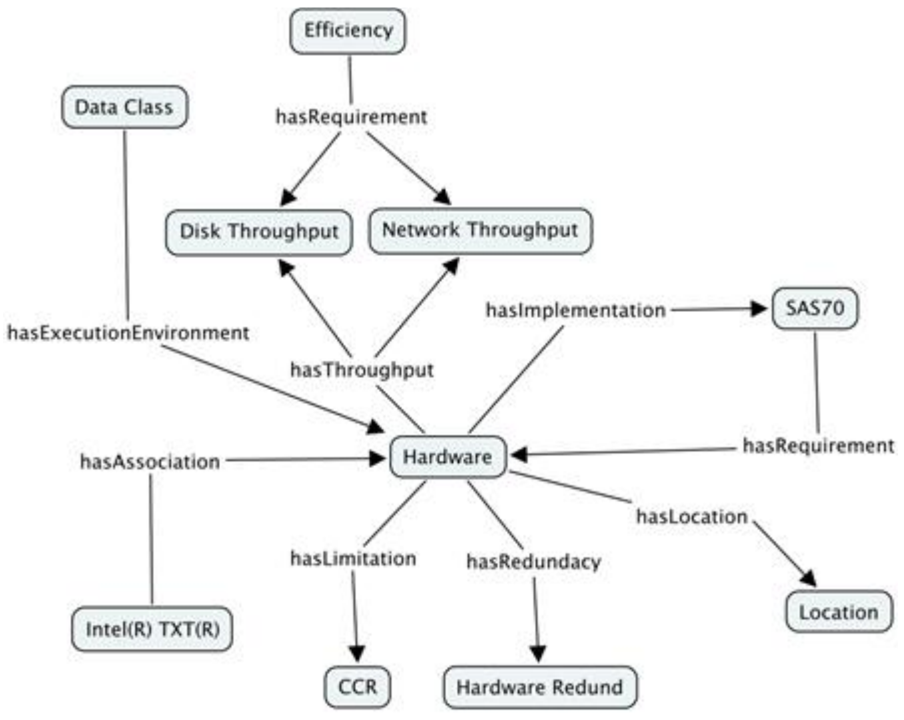


Figure 38: Context Model Part on Hardware

6.2.4 Summary

In this section we have identified the relevant context parameters for the digital preservation of an enterprise business process as specified in the problem statement specified in Section 2.1. To do this, an approach using a combination of top-down and bottom-up methods have been discussed and then carried out. There are benefits as well as limitations to each approach hence why the combination has been used. A bottom-up approach focuses too much on the specific details of a single problem and has the problem of not being generic, whereas the top-down approach looks to divide the enterprise concerns into tangible and manageable segments that can be then more easily presented to managers who traditionally have a hierarchical view of an enterprise. The combination approach is a recommended method that was first suggested in the creation of ontologies by Uschold (Uschold M., et al., 2006) as a middle-out approach.

For the top-down, hierarchical view and for dividing the enterprise we have chosen the Zachman framework as a well tried and tested approach for dividing the concerns that is understandable and whose ideas used currently by many enterprises either directly or indirectly. The Zachman framework guides the division of an enterprise into many perspectives and allows for a holistic view of an enterprise to be captured.

Conversely, for the bottom-up approach, a set of partner-specific scenarios formed the foundation. Scenarios have been designed by all contributing partners individually, and were collaboratively designed in workshops as well as several, regular meetings between the partners involved in this task as well as those partners involved in Task 4.2. These meetings generated several representative scenarios that are important to the different partners. Although the entire set of scenarios identified is not fully captured in the deliverable due to space constraints, they were useful for identifying particular concerns of enterprises that exercise different perpsectives of the Zachman framework. The scenarios do not represent the whole breadth of enterprises but provide a step towards identifying the different areas of concern that can be identified and could be investigated in the scope of TIMBUS. Having multiple scenarios helps exercising both the breadth of the top-down approach as well as the level of detail required for a bottom-up approach.

From the combination of partner-individual contributions, emails, workshops and regular meetings, we have identified a set of contextual parameters that represent the different partners' scenarios. This is a first step in terms of identifying the methodology for generation of contextual parameters from what has been already identified as being a large problem domain. The approach identified in this deliverable will be applied to the more far-reaching and probing use-cases in Workpackages WP7, WP8 and WP9. These larger scenarios will link many of the domains that have been identified so far and will also require certain extensions to what has already been captured to allow for new context parameters that better represent the concerns of those use-cases and allow for more contextual information to be captured. The extensibility of the Context Model is key to its success and the approach taken in this deliverable has allowed for the extension of a structured model based on Zachman to be used. As it grows, the division of concerns will become increasingly important for determining if an adequate representation of the business process has been captured.

6.3 Application to Scenario Processes

The context model developed has subsequently been applied to the scenarios described above. A set of plugins for Protégé has been developed, to allow for a convenient visualization of the individuals and their relations.

6.3.1 Scientific Data Analysis/Scientific Experiment

As the first example, we modelled the scenario of the scientific experiment workflow for music classification, as this scenario was discussed in detail. A graph-representation of the context model is given in Figure 39. The scenario is, as mentioned above, rather focused on technical aspects, thus a significant number of individuals represent data, software components and external systems employed.

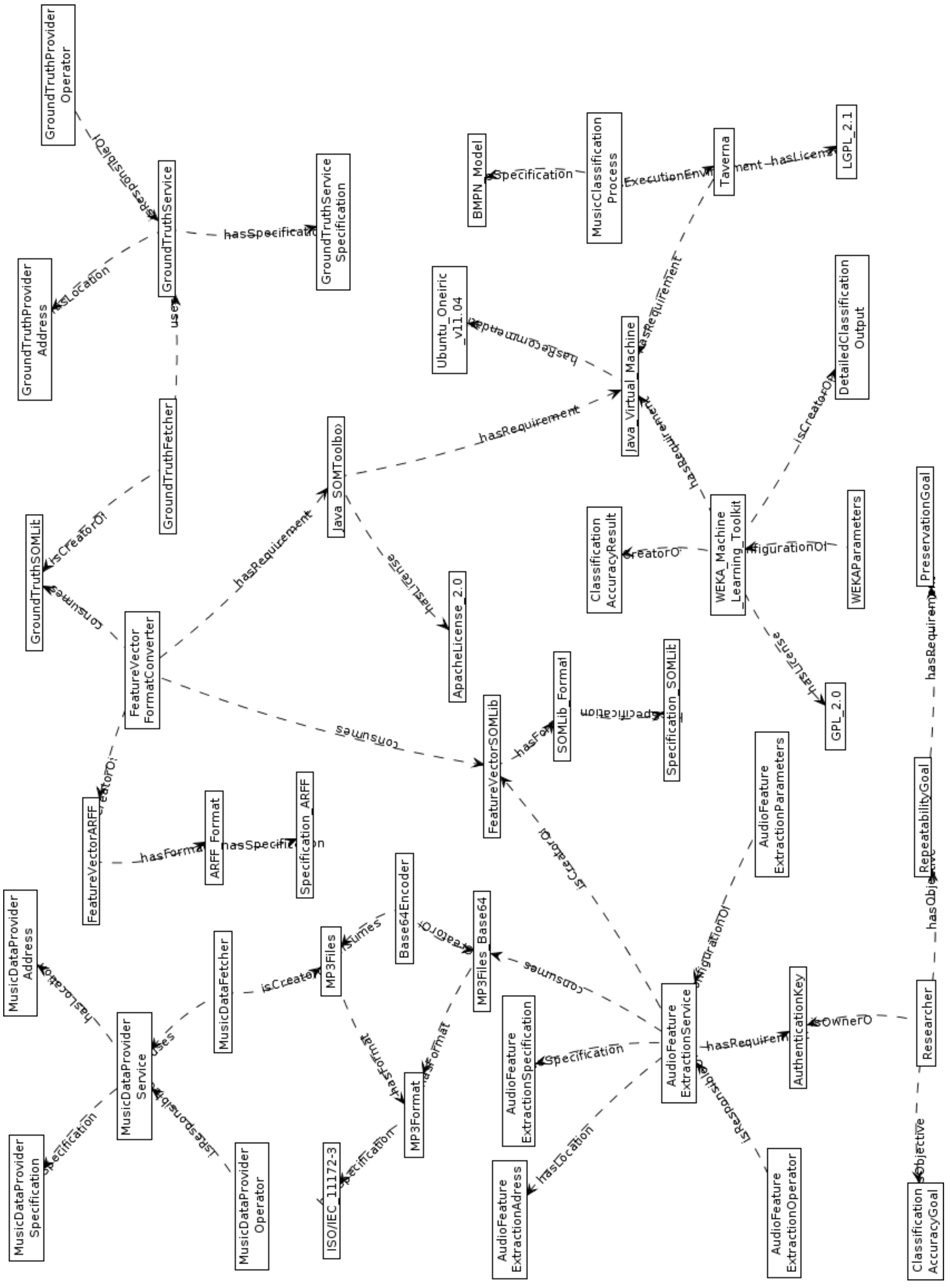


Figure 39: Context Model of Scientific Workflow – Music Classification Experiment

The model illustrates the flow of data between the various components, as well as the data formats and their specifications. This allows determining entry points in the process for capturing data that can be used later for verification purposes.

The various software components are captured with their license information, which plays a role in deciding whether they can be preserved and reused during the exhumation phase at a later stage. External services such as the audio feature extraction are described by their interface definition and location.

However, also the people involved in the process play an important role – their motivation and business plans to provide a service are e.g. important factors that will determine the longevity and availability of their services. The researcher performing the experiment needs to have a certain set of skills, and also owns authorisation information such as personalised license keys to various services.

6.3.2 Knowledge management

The second example covers the knowledge management scenario described above. As in the music classification case, the context ontology includes all complex instances and their object properties. We do not include all data properties as outlined in the context parameters table above to avoid visual clutter. Rather we concentrate on the aspects that influence DP of the IPR protection process as a process associated with a more general knowledge management process. An overview of the corresponding individual graph is given in Figure 40.

Apart from the Zachman perpectives (which are modelled implicitly via the types in the context ontology), the context of the knowledge management scenario may be divided into several broad aspects.

On a high level, (1) there is organisational context, i.e., the interplay between the two companies InnovaCorp and MediaCorp, which has the *MediaServiceAgreement* instance (of type *OrganizationalServiceLevelAgreement*; not shown in the figure) at its core. This agreement is the “bridge” to trigger DP on the part of MediaCorp as soon as DP is indicated on InnovaCorp’s end. On the other hand, at InnovaCorp, the corporate strategy and risks are modelled explicitly in the context in the form of risks and goals, implying the shape of the SLA.

Next, (2) the personnel involved is modelled. As the identity of the persons is essential for the IPR protection, they are represented explicitly in the model: the *Inventor*, *KnowledgeWorker*, *PatentLawyer* and *MediaCreator* nodes.

Moreover, (3) the actual processes involved are modelled. The Knowledge Management process is subdivided into an *IdeaReviewProcess* and a *PatentReviewProcess*, which corresponds to the two stages associated with the knowledge worker and patent lawyer, respectively. Different from the music scenario, however, the processes are not fully structured in the context model with all their steps and transitions as in a BPNM meta-model, which due to their variability would be difficult to model. Rather, the context model covers them as integral representations in BPMN (for the structured part of the knowledge management process) and in appropriate network representations (for the unstructured parts of the patent review process, for instance Markov models or finite state machines). Consequently, a specification of the representations used is required.

Finally, (4) in the case at hand much of the relevant context structure is captured using the technical systems involved, under the assumption that these systems and their associated data cover the structured and unstructured processes up to the depth necessary to replay them after exhumation. This makes the representation of context rely especially on the data of the idea and patent review processes: the *NetWeaverPortalSystem* and its dependent *MediaServiceSystem* need to provide snapshots of their indices (to cover the search processes) and content data (to cover what information was available at the time). Notably, all of the related software is captured to recover these data, and a context capturer may choose to archive the complete systems via a virtualisation approach and/or the particular parts and specifications.

In its current version, the context model relies primarily on OWL individuals that instantiate classes of the generic context ontology, rather than scenario-specific subtypes. While this is expressive enough to model a particular knowledge management process and in particular its related IPR protection processes, for a *set* of similar processes it is more efficient to model a specialised knowledge management context model that contains particular types (as subtypes of the context ontology types) and properties (or sub-properties, with constraints on domain and range, their cardinality etc.).

An example detail in such a model is how the service level agreement may be handled. The instance *MediaServiceAgreement* in the current model (which is of type *OrganizationalServiceLevelAgreement* defined in the generic context ontology) may also be modelled as a scenario-specific sub-type *GeneralMediaServiceLevelAgreement*. On this specific, properties like *hasDeliverer* and *hasRecipient* can be constrained for example to be exactly of cardinality one (“SLAs in media services have only a single deliverer and recipient each“) and also limited to instances of type *Organization*. However, while OWL restrictions are a powerful tool to implement such assertions in sub-ontologies, this is to be taken with a grain of salt as they may imply substantial increase of reasoning complexity, depending on the reasoning profile used, cf. section 3.2.1.2.2.

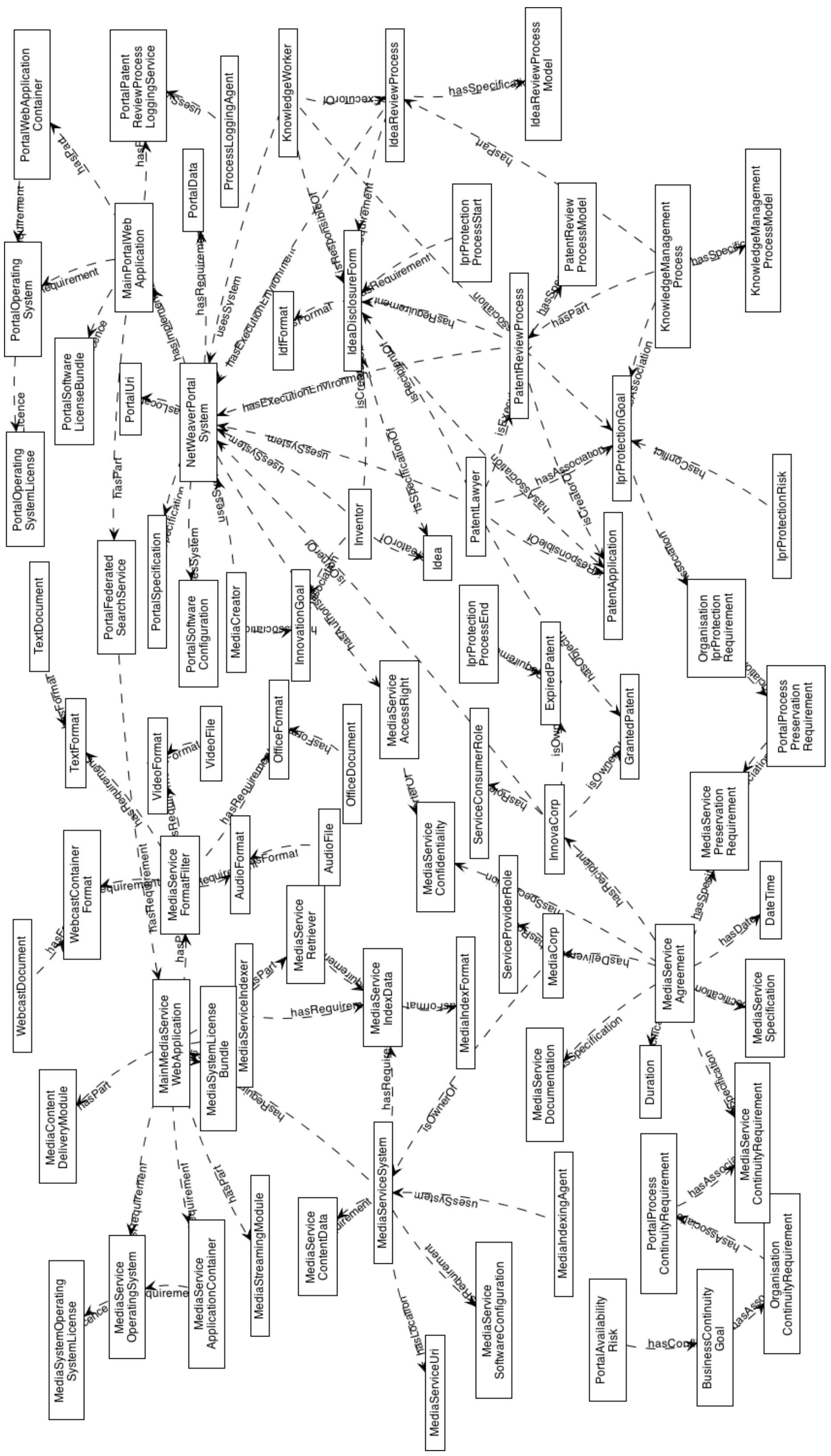


Figure 40: Overview of the Individual Graph of the Knowledge Management Scenario

7 Conclusions and Outlook

Traditional digital preservation approaches focus on preserving digital objects and their context. The context is in the form of Representation Information, which is the information needed so that certain Designated Communities can understand the digital object in the future, and Preservation Description Information, which is the additional preservation metadata needed to manage the preservation of the digital object (Consultative Committee for Space Data Systems, 2002).

In this deliverable we have motivated and illustrated the overall goal of the TIMBUS project to enable successful and feasible digital preservation of entire business processes. This is fostered by capturing the entire relevant context surrounding a business process in an automated or semi-automated way, along with the digital objects that are used by a process and their contexts, so that the future exhumation of the business process is enabled.

This methodology is an expansion of previous digital preservation approaches that presents completely novel challenges. Successful DP of business processes requires capturing sufficient detail of a business process and its context to be able to re-enable its original behaviour at a future date, involving potentially different participating parties, different enabling technologies, different system components (hardware and software), changed services by different service providers (e.g. IoS or SaaS services) or differences in other aspects of the context of a business process.

To explore the relevant context of business processes, this deliverable has covered a comprehensive and scenario-motivated survey on context parameters of business processes which are relevant from TIMBUS' digital preservation perspective. This survey consists of a list of business process use cases that are relevant for digital preservation from a TIMBUS perspective and which informally point out and list context parameters and dependencies between them that are relevant to these scenarios.

Additionally, the formal specification of the TIMBUS Context Model has been based on the results of the survey of relevant context parameters. The model presents a syntactically and semantically unified approach to modelling all relevant context parameters and the dependencies between them around the abstract concept of a business process. In essence, the model provides the foundational framework for process context modelling and reasoning efforts in TIMBUS.

The Context Model enables to represent knowledge (in form of a shared conceptualization) about the "digital preservation of business processes domain". After taking the modelling and reasoning requirements of both deliverables D4.2 and D4.5 and task T6.2 into account, formal ontology has been selected as an appropriate base for the Context Model.

- From a D4.5 perspective we have to model type hierarchies (sub-class relationships between context parameters) and mereological hierarchies (part-of relationships of context parameters and Zachman cells plus perspectives) of relevant context parameters.
- From a D4.2 perspective we have to model additional relations between context parameters.

- Furthermore, expecially when regarding the reasoning requirements of T6.2, we also have to be able to model the required semantics of the relations defined in D4.5 and D4.2.

As these requirements are inherent features of formal ontology, this is an ideal candidate. And as we are modelling certain knowledge (e.g., “all birds can fly”) instead of uncertain knowledge (e.g., “only nearly all birds can fly, as penguins can't”), we focused on ontology formalisms that can model concrete knowledge. In this context, any concrete ontology formalism that can model the required semantics of all required relations (in D4.2 and D4.5) is suitable. Therefore we have selected a standard, well performing (regarding reasoning) and rather expressive one for modelling certain knowledge: the Web Ontology Language (OWL).

In future, deliverable D4.9 will build upon the current state of the Context Model. By adding, removing and modifying represented classes and relations, the model will be iteratively revised. Furthermore, in collaboration with D4.3 and T6.2, we will evaluate whether OWL can keep up with our expressivity expectations (regarding relation semantics) and whether it can keep up with runtime performance expectations.

References

1. Abowd, G. Dey, A Brown, P.J. Davies, N Smith, M and Steggles,P (1999). Towards a Better Understanding of Context and Context-Awareness. In Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing (HUC '99), Hans-Werner Gellersen (Ed.). Springer-Verlag, London, UK, 304-307.
2. Bardram, J. E. (2005). The Java Context Awareness Framework (JCAF) – A Service Infrastructure and Programming Framework for Context-Aware Applications. (A. Gellersen Hans And Want Roy And Schmidt, Ed.)Pervasive Computing, 3468, 98-115. Springer. Retrieved from http://www.springerlink.com/index/yl2fen8clqqwq2tb.pdf
3. Consultative Committee for Space Data Systems (2002) 'Reference Model for an Open Archival Information System (OAIS), Blue Book, Issue 1' , CCSDS - Consultative Committee for Space Data Systems
4. European Parliament (1995) Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data: http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:31995L0046:EN:NOT
5. Gartner (2008) Gartner on Outsourcing, 2008 – 2009. Gartner Research
6. Lankhorst, M (2005) Enterprise Architecture at Work: Modelling, Communication, and Analysis. Springer, 2005.
7. The Library of Congress (2012) PREMIS Data Dictionary: http://www.loc.gov/standards/premis/
8. Matthews, B. McIlwrath, B. Giaretta, D. and Conwa, E. (2008) The Significant Properties of Software: A Study http://www.jisc.ac.uk/media/documents/programmes/preservation/spsoftware_report_redacted.pdf
9. Meeker, H. (2003) “Thinking outside of the Lock Box: Negotiating Technology Escrow”, Computer & Internet Lawyer, No 9
10. Moitra, D. and Ganesh, J (2005) Web services and flexible business processes: towards the adaptive enterprise. Information & Management, 921-933.
11. NCC Group plc (2010) “Preliminary Annual Results for the year ended 31 May 2010”, July 2010
12. NIST (2012) National Software Reference Library: http://www.nsrl.nist.gov/
13. Object Management Group (2011), Business Process Model and Notation (BPMN), Version 2.0, OMG Standard, formal/2011-01-03.
14. The Open Group (2011) TOGAF version 9.1. Van Haren Publishing

15. Object Management Group (2010) Business Motivation Model 1.1. OMG, May 2010.
16. OCLC President (2006) OCLC Presidents letter acknowledging the founding of the PREMIS working group and awards received from the British Library and the society of American Archivists: http://www.oclc.org/uk/en/nextspace/005/letter.htm
17. PREMIS (2005) Final Report of the PREMIS Working Group 2005, with acknowledgements of the original PREMIS members: http://www.oclc.org/research/activities/past/orprojects/pmwg/premis-final.pdf
18. Presutti, V. and Gangemi, A. (2008). Content Ontology Design Patterns as Practical Building Blocks for Web Ontologies. In Proceedings of the 27th International Conference on Conceptual Modeling (ER '08), Qing Li, Stefano Spaccapietra, Eric Yu, and Antoni Olive (Eds.). Springer-Verlag, Berlin, Heidelberg, 128-141.
19. Rabiner, L. R. and Juang, B. H. (1986) An introduction to Hidden Markov Models, IEEE ASSP Magazine: 4 – 16
20. Staab, S. and Studer, R. (2009) Handbook on Ontologies (2nd ed.). Springer Publishing Company, Incorporated.
21. Treinen, R. and Zacchiroli, S. (2008) Description of the CUDF Format
22. US Department of Comerce (2011) US Safe Harbour: http://www.export.gov/safeharbor/
23. van der Aalst, W.M.P. (2011) Process Mining: Discovery, Conformance and Enhancement of Business Processes, Springer-Verlag, Berlin
24. van der Aalst, W. M. P. (2002) Making Work Flow: On the Application of Petri Nets to Business Process Management, Lecture Notes in Computer Science 2360: Application and Theory of Petri Nets, 1-12, Springer-Verlag, Heidelberg-Berlin
25. van der Aalst, W.M.P., Adriansyah, A. and van Dongen, B. F. (2011) Causal Nets: A Modeling Language Tailored towards Process Discovery, CONCUR 2011: 28 – 42
26. Zachman, J (1987) "A framework for information systems architecture," IBM Systems Journal, vol. 12, no. 6, pp. 276–292.
27. Mike Uschold and Michael Gruninger, 2006. Ontologies: Principles, methods and applications. In Journal Knowledge Engineering Review, Vol. 11, pp93—136.