



Initiative for digital transformation in the Metadata and  
Reference Data Sector of the Publications Office of the  
European Union

# Asset Publication Lifecycle Enterprise Architecture

## Disclaimer

The views expressed in this report are purely those of the Author(s) and may not, in any circumstances, be interpreted as stating an official position of the European Union. The European Union does not guarantee the accuracy of the information included in this study, nor does it accept any responsibility for any use thereof. Reference herein to any specific products, specifications, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favouring by the European Union.

This report was prepared for the Publications Office of the European Union by Infeurope.

## Document metadata

<b>Reference</b>	WP 1.1: Asset Publication Lifecycle Enterprise Architecture
<b>Corporate Author</b>	Publications Office of the European Union
<b>Author</b>	Eugeniu Costetchi
<b>Reviewers</b>	Denis Dechandon and Willem Van Gemert
<b>Contractor</b>	Infeurope S.A.
<b>Framework contract</b>	10688/35368
<b>Work package</b>	WP 1.1
<b>Delivery date</b>	25 September 2020
<b>Suggested readers</b>	sector management, business and technical staff, stakeholders, enterprise architects, software developers, project partners

# Abstract

In the European Union, the public sector is one of the most data-intensive sectors. The re-use of the open data can contribute, for example, to the growth of the European economy, the development of artificial intelligence and to overcoming societal challenges.

Given the increasing importance of data standards for the EU institutions, a number of initiatives driven by the public sector, industry and academia have been kick-started in recent years. Some have grown organically, while others are the results of standardisation work. The vocabularies and semantics that they are introducing, together with the technologies that they are using, all differ. These differences hamper data interoperability and thus its reuse by them or by the wider public. This creates the need for a common data standard for publishing public reference data and models, hence allowing data from different sources to be easily accessed and linked, and consequently reused.

The PSI directive across the EU calls for open, unobstructed access to public data in order to improve transparency and to boost innovation via the reuse of public data. The reference data maintained and published by the PO has been identified as data with a high-reuse potential. Therefore, making this data available in machine-readable formats, as well as following the data as a service paradigm, are required in order to maximise its reuse.

In this context, the Publications Office of the European Union maintains and publishes an ever-increasing number of reference data assets vital in the context of inter-institutional information exchange. With regards to reference data, the PO provides an ever-increasing number of services to the main institutional stakeholders and with the aim to extend them to a broader public, enabling active or passive participation in the reference data life cycle, standardisation and harmonisation.

This document provides a working definition of the architectural stance and design decisions that are to be adopted for the asset publication life-cycle process. This process is materialised as the publication workflow and is currently employed by the Standardisation Unit (SU) at the Publications Office of the European Union (PO).

# Contents

<b>1</b>	<b>Introduction</b>	<b>7</b>
1.1	Background considerations . . . . .	7
1.2	EU trajectory towards public sector linked open data . . . . .	8
1.3	Target audience . . . . .	9
1.4	Document scope . . . . .	9
<b>2</b>	<b>Publication workflow digital transformation</b>	<b>11</b>
2.1	State of play . . . . .	11
2.2	Towards semantic technology workflow . . . . .	13
2.3	Towards Service Oriented Architecture (SOA) . . . . .	14
<b>3</b>	<b>Architecture building blocks</b>	<b>15</b>
3.1	Methodology . . . . .	15
3.2	Architecture views . . . . .	16
3.3	ArchiMate elements . . . . .	17
<b>4</b>	<b>Motivation architecture</b>	<b>22</b>
4.1	Overall motivation structure . . . . .	22
4.2	Stakeholders and their roles . . . . .	23
4.3	Drivers: primary . . . . .	25
4.4	Drivers: secondary . . . . .	27
4.5	Assessment: Responsiveness and processing speed . . . . .	28
4.6	Assessment: Data quality . . . . .	29
4.7	Assessment: Service quality . . . . .	30
<b>5</b>	<b>Business use cases</b>	<b>32</b>
5.1	UC1: Evolution: Register request for content change . . . . .	32

5.2	UC2: Evolution: Register request for content change . . . . .	33
5.3	UC3: Implementation: Implement request for content change . . . . .	34
5.4	UC4: Validation: Validate the request for change . . . . .	36
5.5	UC5: Release: Prepare the publication content . . . . .	37
5.6	UC6: Publish: Publish a reference data asset . . . . .	38
5.7	UC7: Publish: Publish a model asset . . . . .	40
<b>6</b>	<b>Business architecture</b>	<b>42</b>
6.1	Prototypical business structure . . . . .	42
6.2	Current asset life-cycle stages . . . . .	43
6.3	Actors and roles . . . . .	45
6.4	Data stewards and custodians . . . . .	46
6.5	Current asset life-cycle overview . . . . .	47
6.6	Current inception and evolution stage . . . . .	49
6.7	Current implementation stage . . . . .	50
6.8	Current pre-release stage . . . . .	51
6.9	Current release stage . . . . .	51
6.10	Current publication stage . . . . .	52
6.11	New asset life-cycle overview . . . . .	54
6.12	New implementation stage . . . . .	55
6.13	New validation stage . . . . .	56
6.14	New release stage . . . . .	56
6.15	New publication stage . . . . .	58
<b>7</b>	<b>Application architecture</b>	<b>60</b>
7.1	Prototypical application structure . . . . .	60
7.2	Current and new application service architecture . . . . .	62
7.3	Inception and evolution services and components . . . . .	64
7.4	Implementation services and components . . . . .	65
7.5	Pre-release and validation services and components . . . . .	69
7.6	Release services and components . . . . .	70
7.7	Publication services and components . . . . .	72
<b>8</b>	<b>Technology architecture</b>	<b>75</b>
8.1	Prototypical technology structure . . . . .	75
8.2	Current technology architecture . . . . .	76
8.3	New technology architecture . . . . .	78

<b>9</b>	<b>Technology deployment proposal</b>	<b>80</b>
9.1	VocBench3 export and automation services . . . . .	80
9.2	Validation services . . . . .	82
9.3	Transformation services . . . . .	83
9.4	Reporting services . . . . .	85
9.5	Service orchestration . . . . .	85
<b>10</b>	<b>Conclusions</b>	<b>88</b>
10.1	Summary . . . . .	89
10.2	Limitations and future work . . . . .	89
10.3	Final word . . . . .	90

# Chapter 1

## Introduction

This document provides a working definition of the architectural stance and design decisions that are to be adopted for the asset publication life-cycle process. This process is materialised as the publication workflow and is currently employed by the Standardisation Unit (SU) at the Publications Office of the European Union (PO).

In this document we (a) establish the baseline architecture, supported by strategic and motivational information; and (b) develop a target architecture guiding the digital transformation processes towards new technologies. This constitutes a natural evolution in response to changing mission needs defined by SU management, and also takes into consideration the strategic directions proposed by the PO, the European Commission (EC) and the European Parliament (EP) as presented below.

### 1.1 Background considerations

Given the increasing importance of data standards for the EU institutions, a number of initiatives driven by the public sector, industry and academia have been kick-started in recent years. Some have grown organically, while others are the results of standardisation work. The vocabularies and semantics that they are introducing, together with the technologies that they are using, all differ. These differences hamper data interoperability and thus its reuse by them or by the wider public. This creates the need for a common data standard for publishing public reference data and models, hence allowing data from different sources to be easily accessed and linked, and consequently reused.

The PSI directive [21] across the EU calls for open, unobstructed access to public data in order to improve transparency and to boost innovation via the reuse of public data. The reference data maintained and published by the PO has been identified as data with a high-reuse potential [3]. Therefore, making this data available in machine-readable formats, as well as following the data as a service paradigm, are required in order to maximise its reuse.

In this context, the Publications Office of the European Union maintains and publishes an ever-increasing number of reference data assets vital in the context of inter-institutional information exchange. With regards to reference data, the PO provides an ever-increasing number of services to the main institutional stakeholders and with the aim to extend them to a broader public, enabling active or passive participation in the reference data life cycle, standardisation and harmonisation.

## **1.2 EU trajectory towards public sector linked open data**

European institutions started out to adopt Semantic Web and Linked Data technologies as part of their visions to become data-centred e-government bodies [17, 19].

The EU institutions also aim for implementation of a single digital gateway to “facilitate interactions between citizens and businesses, on the one hand, and competent authorities, on the other hand, by providing access to online solutions, facilitating the day-to-day activities of citizens and businesses and minimising the obstacles encountered in the internal market. The existence of a single digital gateway providing online access to accurate and up-to-date information, to procedures and to assistance and problem-solving services could help raise the users’ awareness of the different existing online services and could save them time and expense” [20]. This is well in line with earlier established goals for encouraging the open data and the re-use of public sector information [18, 21].

Many of the legacy systems used in the EU institutions use XML data format for exchange and document formats governed by the XSD schemes [41]. The aim is to evolve so that both existing and new systems are capable to operate with semantic data representations using RDF [55], OWL [40, 38], SHACL [30] and other representations, and serialised at least in RDF/XML [6, 42], Turtle [10] and JSON-LD [46, 45] formats.



For this reason, the PO has already been publishing data in RDF format for over a decade using the Cellar repository [22]. Also, the SU, in particular, is committed to publish and disseminate reference data in semantic formats.

## 1.3 Target audience

The target audience for this document comprises the following groups and stakeholders:

- Management of the SU
- Enterprise architects and data governance specialists
- Documentalists involved in the reference data life-cycle
- Technical staff in charge of operating workflow components
- Developers in charge of workflow and component implementation
- Third parties using the SU services and data in charge of harmonisation and standardisation of metadata and processes

## 1.4 Document scope

This document aims to support SU in the transition towards semantic technologies with a particular focus on the architecture of the publication workflow.

The central use case is to support the asset publication life-cycle detailed in Section 6.2. This includes managing the incoming requests, editing the reference assets in VocBench3 system [47, 48], then exporting the RDF data and passing them as input to a set of processes that validate, assess, transform, package and finally publish the assets in Cellar [22], the main dissemination platform. This use case has been broken down into sub- use cases that are detailed in Section 5.

This document will provide a motivational, business and application account of the asset life-cycle workflow. Each of these accounts is limited strictly to the success scenario of the above-mentioned use case and does not include possible extensions and variations.

There is a series of aspects that were intentionally left out. For example the recommendations related to the data governance both internally within the SU and also externally in relation with partners, stakeholders and clients. No implementation details are specified for the new components. This shall be covered elsewhere. Little or not account is provided about the data structures and static objects used in the business process or exchanged between the application services. No monitoring or performance measurement system is foreseen by this architecture across all levels starting from motivation level key performance indicators (KPI), through business level process monitoring, down to performance measurement of the applications and the infrastructure indicators.

This architecture is also focused on the transition from XML-base asset sources representation to the RDF-based representation along with the resulting implications. And, for instance, does not address in great detail how the new impact assessment module shall be implemented; or how the workflow orchestration engine shall be configured, what process automation service to use, or what technologies could be chosen for that. Such decisions shall be carried out in subsequent steps. No asset inventory and metadata management system is proposed either, it is only identified as missing and necessary in Section 7.4.

With this scope in mind, we present in the next section, what is the general direction that the digital transformation shall take and what is the starting point for it, given the business and technical state of play.

## Chapter 2

# Publication workflow digital transformation

Internally in the SU publication workflow is called the entire process from receiving the request for change of a data asset, through its implementation, validation, transformation of data, packaging and dissemination, ending finally with the data being consumed by the client, who request new changes again after using the data. We refer to this publication workflow as the publication life-cycle process, or the asset life-cycle process.

In this chapter we assess the technical state of play in order to be able to frame technically the digital transformation proposed in this architecture. In the next section (Section 5) we will address the business aspects in terms of use case scenarios, and then, in Section 6 in terms of business processes, which will serve as a backbone for organising the application architecture in Section 7.

### 2.1 State of play

The SU publishes reference data in several formats, the most important being XML [8], XSD [41] and RDF/XML [6, 42]. On the application side (see Section 7), the SU currently employs a legacy (custom-built) system for controlling and executing, in part, the asset management life-cycle operations (legacy workflow system). The system was developed using a mixture of XSLT technology [27], Perl and Bash scripting languages. The system was developed to execute a wide variety of conversions

and transformations based on XML source files into various other formats including human readable documents.

The source data representation (XML in this case) has the primary role to serve as the original reference (and the only source of truth) and, additionally, maintaining non-redundancy and rich expressivity<sup>1</sup>. All other data forms and representations are secondary and are generated by transformation and conversion processes from the source representation.

One peculiarity of the legacy system setup is that the editing of the asset content is performed using MS Excel [34]. This is done by transformation of the content from XML representation into Excel style-sheets, which are edited by SU documentalists and then converted back into XML format. In this way, a circular transformation is achieved which also serves as an integrity checking and validation mechanism. In addition, XSD [41] schema definitions are used to validate the XML source representation.

The legacy workflow system uses the file system for data persistence. In addition, this functionality is aided by a version controlling system, SVN [2], to trace the evolution of data across time.

Some steps in the legacy workflow have been automated. The automation is based on cron jobs<sup>2</sup> and SVN hooks that, upon changes in the source XML or Ms Excel files, trigger a set of conversion mechanisms. Some other steps require manual triggering and eventually parameterisation intervention. The execution of the automated steps often requires intervention by technical staff or someone with above-average IT skills which represent an impediment for the non-technical documentalists and a hindrance for the IT staff.

Moreover, the maintenance of this system is burdened by a technical debt that has accumulated over time, because the system has evolved organically based on incorporation of many requests.

---

<sup>1</sup>In computer science, the expressive power (also called expressiveness or expressivity) of a language is the breadth of ideas that can be represented and communicated in that language. The more expressive a language is, the greater the variety and quantity of ideas it can be used to represent.

<sup>2</sup>The software utility cron, also known as cron job, is a time-based job scheduler in a Unix-like computer operating systems

## 2.2 Towards semantic technology workflow

The SU's mission regarding the technological evolution is to migrate towards Semantic Web and Linked Data technologies and representations. The maintenance of reference data is currently done based on XML source representation and the desired transition is towards RDF-based representation [55, 39]. For that purpose, MS Excel and XML sources are no longer suitable and a dedicated editor is necessary.

To solve this issue, SU took the development flagship of the VocBench3 [48, 47] system, a web-based, multilingual, vocabulary editing tool based on the SKOS [36] model, which is modelled with RDFS [24, 9]. Later, VocBench3 was developed to support authoring of RDFS [9] and OWL [40] vocabularies.

Switching to RDF-based sources and adoption of VocBench3 system implies a technological and business process disruption. The main reason for this is that the legacy workflow system operates only with XML-based sources and does not support RDF sources. RDF representation being only a by-product derived from XML.

VocBench3 naturally adopted a persistence based on triple stores, which are NoSQL<sup>3</sup> database systems implementing the directed graph data model instead of the hierarchical or relational data model. The relational data model is mentioned here because the MS Excel worksheets are based on tabular data organisation; the hierarchical data model is mentioned because XML is fundamentally a hierarchically organised data structure; also, both are only partially compatible with the graph paradigm present in semantic data models [25].

Migration towards a new workflow that integrates VocBench3 requires reconciliation between file-system and database approaches to persistence. Also, a paradigmatic transition to graph-based data representation in RDF, from the hierarchical models of source representation in XML, and tabular models used for source authoring in Excel, is necessary.

The legacy workflow system is also lacking in semantic validation, structural analysis (fingerprinting) and content comparison capabilities (calculating the difference between two versions of an asset) for RDF data representation. A transition would imply development of at least these new capabilities in order to maintain business processes similar to the current ones (see Section 6).

---

<sup>3</sup>A NoSQL database provides a mechanism for storage and retrieval of data that is modelled in means other than the tabular relations used in relational databases.

## 2.3 Towards Service Oriented Architecture (SOA)

*Service-oriented architecture* (SOA) [23] is a style of software design where services are provided to the other components by application components, through a communication protocol over a network. A SOA service is a discrete unit of functionality that can be accessed remotely and acted upon and updated independently. SOA is also intended to be independent of vendors, products and technologies.

SOA is believed to help businesses respond more quickly and more cost-effectively to changing market conditions. This style of architecture promotes high component reuse and facilitates interconnection between components. The SOA systems are known to be resilient in the light of changing technologies because the components are loosely coupled and could easily be replaced by alternative implementations exposing with the same interfaces. SOA could be regarded as an architectural evolution rather than as a revolution. It captures many of the best practices of previous software architectures and it is a practical approach for cloud computing [51].

Service-oriented architecture can be implemented with web services or Micro-services [7]. This is done to make the functional building-blocks accessible over standard Internet protocols that are independent of platforms and programming languages. These services can represent either new applications or just wrappers around existing legacy systems to make them network-enabled [11]. We recommend this approach for the transition to a micro-service based application. First the micro-services can be implemented as wrappers around existent components and at a later stage the components can be replaced by modern implementations.

# Chapter 3

## Architecture building blocks

This chapter provides some foundations about the notation, definitions the general approach adopted here to model the enterprise architecture.

### 3.1 Methodology

In this document we take an enterprise architecture perspective and aim to provide several architecture views (see Section 3.2) which are necessary and sufficient to describe the asset lifecycle process.

In developing this architecture, we are in part using the TOGAF [49] methodology, which is, in fact, a framework for enterprise architecture that provides an approach for designing, planning, implementing and governing an enterprise information technology architecture. Although we do not follow this framework completely, we took inspiration from parts of it which were applicable to the goals of this architecture.

For architecture representation, we adopt ArchiMate language [50], which is an open and independent enterprise architecture modelling language to support the description, analysis and visualisation of architecture within and across business domains in a clear and unambiguous way.

We have conducted a series of interview with the SU management, the technical team and the business teams. In developing the motivation architecture, we entirely rely on the input from the SU management, presented in Section 4.

The business use cases represent the knowledge elicited from the technical team and the business teams and are presented in Section 5.

The other layers of this architecture, are a gradual fleshing out of the use cases, and rely on the author experience of working a few years side by side with SU business (documentalists) and technical teams. This experience results in the knowledge of the applications and technical peculiarities of the SU.

The corresponding ArchiMate diagrams were modelled and designed using Enterprise Architect Tool [44]. Finally this report was written covering the overall architecture.

## **3.2 Architecture views**

Architecture views are an ideal mechanism to purposefully convey information about architecture areas. In general, a view is defined as a part of an Architecture Description that addresses a set of related concerns and is tailored for specific stakeholders. A view is specified by means of an architecture viewpoint, which prescribes the concepts, models, analysis techniques, and visualisations that are provided by the view. Simply put, a view is what you see, and a viewpoint is where you are looking from [50].

An architecture view expresses the architecture of the system of interest in accordance with an architecture viewpoint (or simply “viewpoint”). There are two aspects to a viewpoint: the concerns it frames for the stakeholders and the conventions it establishes on views [50].

Viewpoints are designed for the purpose of communicating certain aspects and layers of an architecture. In this document we address the motivation view (Section 4), the business view (Section 6), the application view (Section 7) and the technology view (Section 8).

Instead of describing what each of these views represents in this section, we decided to provide such an description in the beginning of each of the subsequent sections. This way, we aim to ease the section reading by providing the reader a fresh introduction into the structure of a prototypical layer architecture before the actual SU architecture is described.



### 3.3 ArchiMate elements

This section presents the ArchiMate elements, in terms of their definition and the graphical notation, which we employ in each of the architecture views.

Table 3.1: Overview of the relevant motivation elements [50]



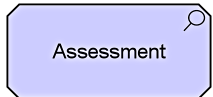

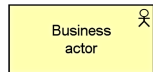



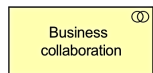

Element	Definition	Notation
Stakeholder	Represents the role of an individual, team, or organisation (or classes thereof) that represents their interests in the effects of the architecture.	
Driver	Represents an external or internal condition that motivates an organisation to define its goals and implement the changes necessary to achieve them.	
Assessment	Represents the result of an analysis of the state of affairs of the enterprise with respect to some driver.	
Goal	Represents a high-level statement of intent, direction, or desired end state for an organization and its stakeholders.	

Table 3.2: Overview of the relevant business layer elements [50]

Element	Definition	Notation
Business actor	Represents a business entity that is capable of performing behaviour.	 
Business role	Represents the responsibility for performing specific behaviour, to which an actor can be assigned, or the part an actor plays in a particular action or event.	 
Business collaboration	Represents an aggregate of two or more business internal active structure elements that work together to perform collective behaviour.	 

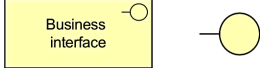


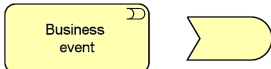

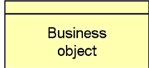

<i>Table 3.2 continued from previous page</i>		
Business interface	Represents a point of access where a business service is made available to the environment.	
Business process	Represents a sequence of business behaviours that achieves a specific result such as a defined set of products or business services.	
Business function	Represents a collection of business behaviour based on a chosen set of criteria (typically required business resources and/or competencies), closely aligned to an organisation, but not necessarily explicitly governed by the organisation.	
Business event	Represents an organisational state change.	
Business service	Represents explicitly defined behaviour that a business role, business actor, or business collaboration exposes to its environment.	
Business object	Represents a concept used within a particular business domain.	
Representation	Represents a perceptible form of the information carried by a business object.	

Table 3.3: Overview of the relevant application layer elements [50]

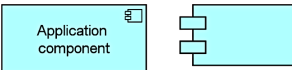
Element	Definition	Notation
Application component	Represents an encapsulation of application functionality aligned to implementation structure, which is modular and replaceable.	

Table 3.3 continued from previous page











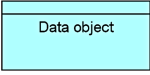
Application interface	Represents a point of access where application services are made available to a user, another application component, or a node.		
Application function	Represents automated behaviour that can be performed by an application component.		
Application process	Represents a sequence of application behaviours that achieves a specific result.		
Application event	Represents an application state change.		
Application service	Represents an explicitly defined exposed application behaviour.		
Data object	Represents data structured for automated processing.		

Table 3.4: Overview of the relevant technology layer elements [50]

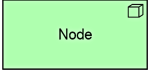

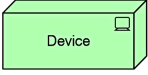

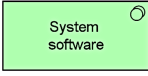



Element	Definition	Notation	
Node	Represents a computational or physical resource that hosts, manipulates, or interacts with other computational or physical resources.		
Device	Represents a physical IT resource upon which system software and artefacts may be stored or deployed for execution.		
System software	Represents software that provides or contributes to an environment for storing, executing, and using software or data deployed within it.		
Technology interface	Represents a point of access where technology services offered by a node can be accessed.		

Table 3.4 continued from previous page







Communication network	Represents a set of structures that connects nodes for transmission, routing, and reception of data.		
Technology service	Represents an explicitly defined exposed technology behaviour.		
Artefact	Represents a piece of data that is used or produced in a software development process, or by deployment and operation of an IT system.		

Table 3.5: Overview of the ArchiMate relationships [50]


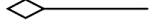
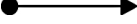
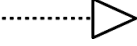

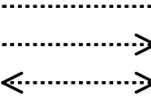
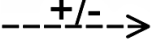
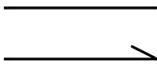


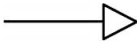


Element	Definition	Notation
<b>Structural Relationships</b>		
Composition	Represents that an element consists of one or more other concepts.	
Aggregation	Represents that an element combines one or more other concepts.	
Assignment	Represents the allocation of responsibility, performance of behaviour, storage, or execution.	
Realisation	Represents that an entity plays a critical role in the creation, achievement, sustenance, or operation of a more abstract entity.	
<b>Dependency Relationships</b>		
Serving	Represents that an element provides its functionality to another element.	
Access	Represents the ability of behaviour and active structure elements to observe or act upon passive structure elements.	
Influence	Represents that an element affects the implementation or achievement of some motivation element.	

Table 3.5 continued from previous page		
Association	Represents an unspecified relationship, or one that is not represented by another ArchiMate relationship.	
Dynamic Relationships		
Triggering	Represents a temporal or causal relationship between elements.	
Flow	Represents transfer from one element to another.	
Other Relationships		
Specialisation	Represents that an element is a particular kind of another element.	
Junction	Used to connect relationships of the same type.	<div> (And) Junction  Or Junction</div>

# Chapter 4

## Motivation architecture

This chapter presents the motivation and goal structure of the Standardisation Unit (SU). This helps to scope and derive a rationale for the current architecture specification.

We do not aim for an in depth coverage of the motivation architecture here. So it cannot be considered as a fully fledged decision-making tool for the management. What we rather aim at is accounting for the context of the asset publishing lifecycle which is the final goal of this architecture.

Nonetheless, this motivation view helps address questions on why a demand is meaningful, model crucial drivers and root causes behind the demand, actual goals and related outcomes, as well as concrete requirements for further development. In short, it answers the crucial questions to WHOM, WHY and WHAT.

### 4.1 Overall motivation structure

The structure of motivations, in ArchiMate, is organised hierarchically in several layers. For simplicity, we have chosen to use the top four layers: *stakeholders*, *drivers*, *assessments and goals*; leaving out the *outcomes*, *principles* and *requirements*. Figure 4.1 depicts the organisation of the motivation architecture. The structure starts at the top with enumerating the stakeholders, who can be individuals, teams or organisations that represent their interests in the effects of the architecture [50].

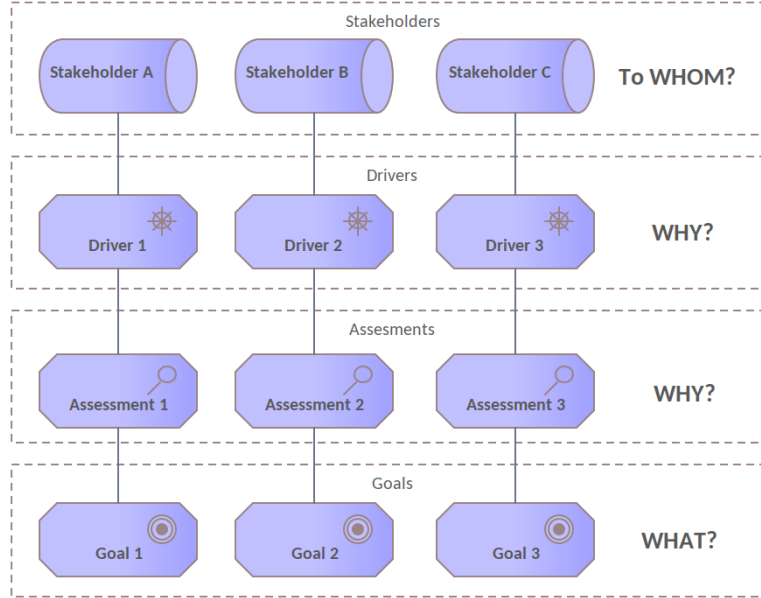


Figure 4.1: The layered motivation structure

*Stakeholders* have associated interests, concerns or drivers, which represent internal or external conditions that motivate an organisation to define goals [50].

*Assessments* represent results of analysis of the state of affairs with respect to a driver. They reveal strengths and weaknesses, opportunities or threats to an area of interest [50].

Assessments are associated with goals which represent a high-level statement of intent, plus direction to desired end state for an organisation and its stakeholders [50].

Next we present the SU motivation structure spread over several sections.

## 4.2 Stakeholders and their roles

The Standardisation Unit involves multiple stakeholders. We could enumerate them; however, the list would be long and is outside the scope of this exercise. Instead, we highlight the most important ones and, in addition, we group them based on the role they play in interaction with SU. In Figure 4.2 the roles are depicted as aggregate

stakeholders in a grouping frame in the middle of the figure. Above the roles are positioned the most important external stakeholders, while below the stakeholders from the PO are enumerated.

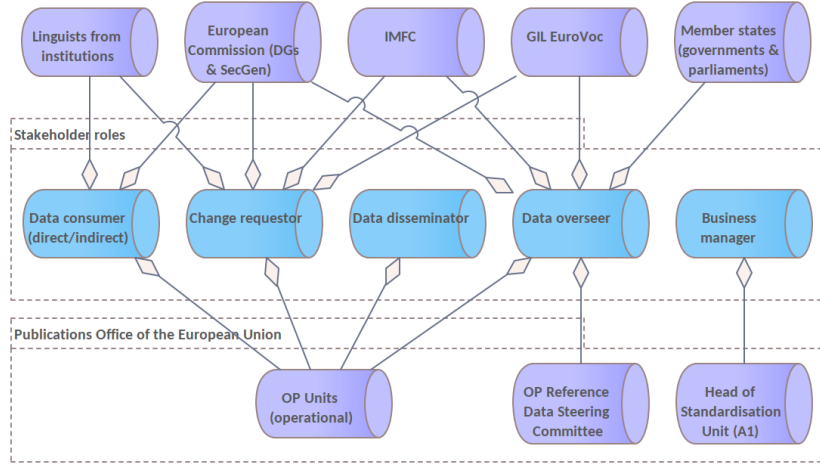


Figure 4.2: The layered motivation structure

The most important external stakeholders are: the European Commission, together with the Secretary General and all of the Commission’s Directorates, the Inter-institutional Metadata and Formats committee (IMFC), the EuroVoc Committee (Group inter-institutional Lex (GIL)-subgroup EuroVoc), EU Member States (MS) represented by their governments and parliaments, and linguists from different institutions and with a particular interest to the IATE project.

In Figure 4.2, the OP stakeholders are placed in organisation-bound context, as the SU is a part of the PO and so its sibling units are not entirely external but are members of the same organisation. The stakeholders within the PO are the various units that use the reference data (e.g. Cellar team, OP Portal team, EurLex team etc.). A Reference Data Steering Committee is scheduled to be formed in the near future in order to coordinate and harmonise the published reference data. And, finally, the Head of the Standardisation Unit (SU) who is in charge of running the enterprise, is also a stakeholder.

In order to more easily account for the stakeholders’ drivers, interests and goals, we grouped them based on their roles in interaction with SU. In Figure 4.2, the roles are depicted as aggregate stakeholders in a grouping frame in the middle of the



figure. The roles include: data consumers, data requester, data disseminators, data overseers and business managers.

Next we describe each of the roles and briefly enumerate the stakeholders.

The *consumers* of assets are users who directly engage with published assets (direct consumers) and can also be users of applications and services that are making use of the published assets (indirect consumers).

The *change requester* are agents who require particular content to be available as reference data.

The *data disseminators* are services and platforms where reference assets are published for broad public consumption.

The *data overseers* are agents who ensure that content satisfying business needs is harmonised, coherent and complete. They are also responsible for the content correctness and harmonisation among multiple stakeholders and its usefulness in the broader context of application. Usually the role of data overseers is performed by the standardisation committees, steering committees and data stewards at large.

### 4.3 Drivers: primary

We have identified four *primary drivers*, three *secondary drivers* and two *internal efficiency drivers*. The distinction between primary and secondary drivers is based on whether the driver is shared, or not, between the external and internal stakeholders (in this case the business management). Figure 4.3 depicts the main stakeholders and their concerns, where the business manager, in this case head of the SU, has the same primary concerns as the main stakeholder roles.

For change requesters, the interaction *responsiveness and the speed of the asset life-cycle process* is of primary concern. The sooner the requests are processed and analysed, the sooner they can be implemented, processed and published. The goal of the SU is to rapidly publish overnight change requests, as compared to the current situation when four major publications are scheduled per year allowing also a few urgent ones in between.

The *quality of service* provided by the SU at large, and service consumer satisfaction, is a direct concern for change requesters and data overseers as primary users of various SU services.

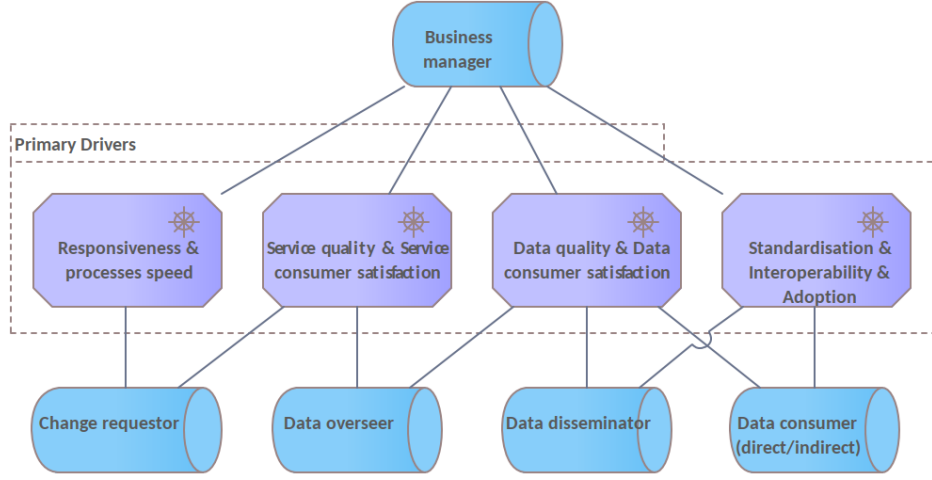


Figure 4.3: Primary drivers, motivating both, the internal and external stakeholders

The *quality of data* is of special interest for data overseers as they are directly responsible for this aspect and implicitly of the data consumer satisfaction. The data quality here has a wide meaning covering aspects of formal, semantic and conceptual correctness while also being timely and up-to-date with the business. Besides data overseers, the data users are also interested in high-quality reference data. Data disseminators are indirectly affected by the quality of the data they distribute and also share this interest to a lesser degree.

The last of the primary drivers is *standardisation, interoperability and adoption*, which is a major concern for data disseminators and data users. This driver covers the adoption of widely-used meta-models, formally well-defined models representing shared conceptualisation of major bodies and organisations, usage and implementation of national and international standards proposed by standardisation bodies (e.g. ISO, W3C, OMG). These standards refer not only to aspects of data representation, but also to protocols, exchange schemas, validation mechanisms and other tools facilitating systemic interoperability.

## 4.4 Drivers: secondary

The secondary drivers are those that are important to both external and internal stakeholders. They are depicted in Figure 4.4.

*Steady and uninterrupted IT systems* is a critical driver for the data consumers. Several institutions and OP units have built IT systems which rely on data maintained and published by the SU. Also, this is one dimension of client satisfaction. The SU ensures that data modifications have no impact on external systems, and for that a part of the life-cycle process is dedicated to impact assessment.

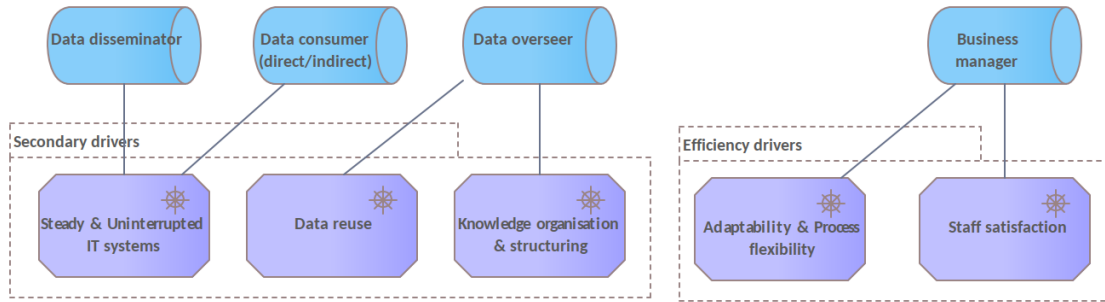


Figure 4.4: Secondary drivers, motivating either internal or external stakeholders

*Data reuse* is encouraged at the level of all EU institutions as a means to reduce integration and development costs. The standardisation committees and other responsible parties (the data overseers) are especially interested in this.

Keeping the knowledge organisation and its structure in a coherent and consistent form is not a trivial task. Reaching a common shared conceptualisation over a given domain is a goal that is difficult to achieve. Nevertheless, it is a precondition if the data is to be reused by multiple parties. Therefore, this constitutes another driver for data overseers.

Internally, *staff satisfaction* is an important driver for the SU management. Making sure the business and technical teams can fulfil their duties in a flawless and unhindered manner impacts directly their engagement, satisfaction and productivity.

Requests from external clients sometimes do not lead to data changes alone, but result in changes of the technical processes as well, if not developments of additional processes and components. Adapting and configuring the currently-used process is becoming increasingly difficult; therefore, making sure that these operations are

possible with minimal effort and that the production process is flexible, represents a driver for SU management.

Next we have chosen three drivers we considered to be the most important for this exercise, and we present their assessments.

## 4.5 Assessment: Responsiveness and processing speed

A number of causes were identified that decrease responsiveness and process speed.

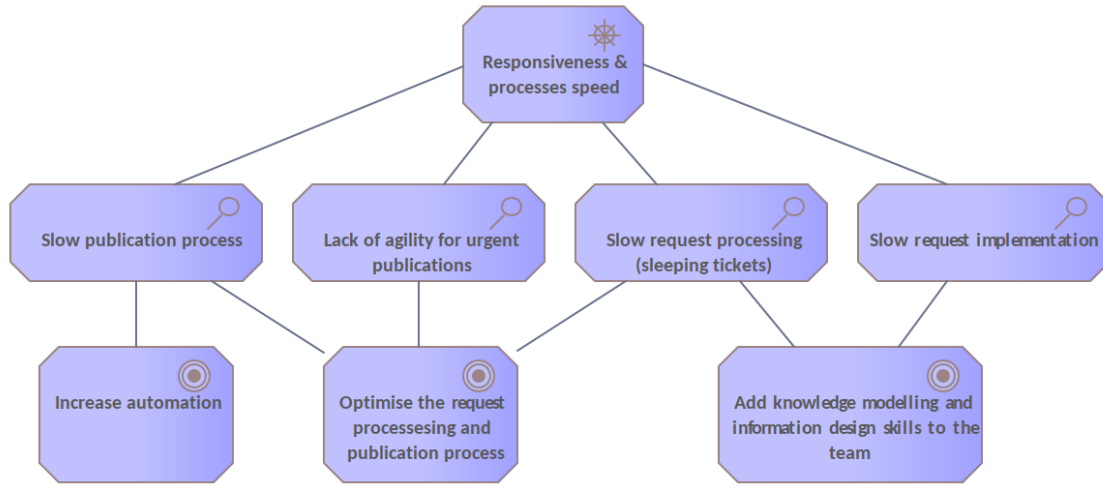


Figure 4.5: The assessment of the responsiveness and processing speed driver

First of all, execution of the entire publication process (including technical and business processes) is slow. This includes burdening by urgent publication requests, which constitute a technical limitation to processing one asset per day.

To address these problems, we aim to increase the automation level which mostly has an impact on technical aspect of the workflow processes. In addition, the increase of automation level is also to optimise the current workflow process, mostly regarding business aspects, in order to reduce the bottlenecks and streamline production.

In addition, slow ticket processing can be due to a limitation in the number of tickets that can be handled by the business team: a caveat here is the lack of modelling and

knowledge organisation kills. As a result, training the entire team to add knowledge modelling skills or recruiting additional staff with such skills are potential solutions.

## 4.6 Assessment: Data quality

Data quality and, implicitly, consumer satisfaction driver have been assessed with the following issues being identified.

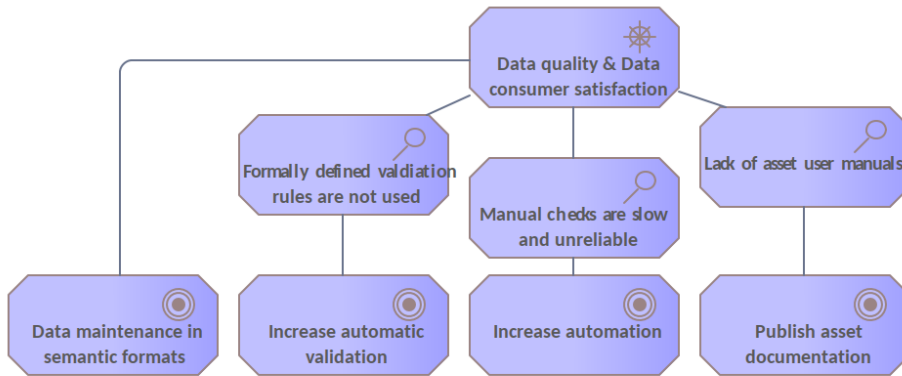


Figure 4.6: The assessment of data quality and data consumer satisfaction

The published assets are difficult to adapt by new consumers due to the lack of user manuals. Requests come for both documentation on the conceptual organisation of the asset and the data models used for representing it. We recommend that complete documentation is developed and then published for all assets.

At least for the RDF part of the current publication workflow, the quality assessment is done by technicians undertaking manual checks, ensuring that the transformation processes ran correctly. Therefore, for example, preparing a new version of EuroVoc thesaurus takes days to complete. Moreover, formally-defined validation rules are minimally employed, if at all. The RDF-related processes do not employ SHACL validation, even if the SHACL shape definitions are available. On the XML side, non-semantic validation rules are possible. For these two reasons we propose, as in the previous section, to increase automation and to start employing increasingly more automated validation in order to reduce the time spent by SU team members verifying content before publication.

Finally, the switch from the XML-based asset sources to RDF-based asset sources

is highly recommended as data maintenance is considerably improved in semantic formats.

## 4.7 Assessment: Service quality

A number of complaints have been received, mainly from other OP units, with regards to the quality of services provided by the SU.

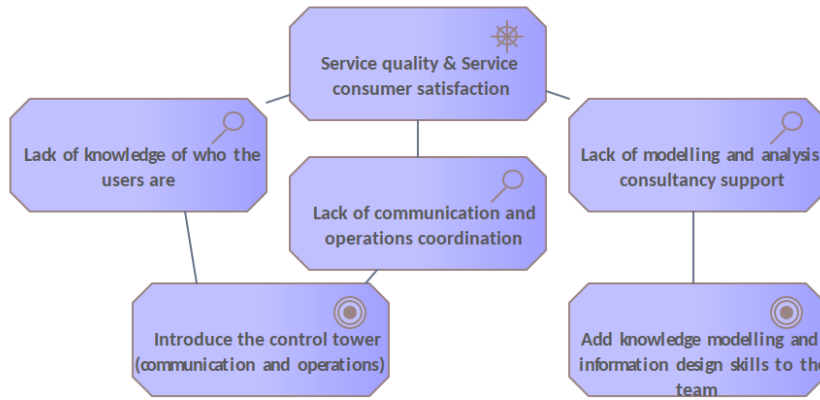


Figure 4.7: The assessment of service quality and service consumer satisfaction

The complaints are mainly due to the lack of OP internal synchronisation between units, both regarding communication and operational coordination. This is in part linked to a lack of precise knowledge of who the users are and in which manner they engage with assets published by the SU.

To address these two issues, our main recommendation (aligned with SU management vision) is to introduce a control tower-like faculty into the SU, which would take care of synchronisation (communication-wise and operational) with all OP units, and if possible with parties from other institutions and Member States.

Another request often received by the SU is to provide modelling and knowledge organisation services to third party clients: the SU does not currently offer such service. In order to establish the grounds for setting up such service provision in the future, we recommend training current members on knowledge modelling and information design topics, or eventually employing additional staff with such skills.

This brings us to the end of the motivation structure section. While we did not provide an extensive account, we did however identify some SU motivations, drivers and goals that are relevant, directly or indirectly, to the aims of this enterprise architecture.

# Chapter 5

## Business use cases

This chapter presents the core business use cases that have been identified in discussions with SU team members. These use cases have been structured in the light of the new asset life-cycle process, and not the current one, even though they are heavily inspired by the current process. Section 6 presents the designs of current and new asset life-cycle processes, illustrating both differences and commonalities. The designed processes are derived from the use case descriptions presented below.

### 5.1 UC1: Evolution: Register request for content change

**Trigger:** A request for a content change is received in the functional mailbox.

**Success guarantee:** A complete and clear change request case is created and scheduled for approval.

**Main success scenario:**

1. The client requests a change of one or multiple concepts in an asset
2. The request manager creates a new change request case and acknowledges the client
3. The asset manager analyses the request (in terms of business needs and data management implications) and summarises the case



4. The request manager informs the client of the case summary
5. The request manager proposes the case for discussion in the next meeting of the team steering committee

**Extensions:**

- 4a The request is incomplete or unclear:
  - 4a1 The asset manager formulates the information requirements
  - 4a2 The request manager collects the required details and clarifications from the client
  - 4a3 Return to step 3
- 4b The request is complex and needs deeper conceptual analysis and modelling/design:
  - 4b1 The asset manager presents the case to the knowledge modelling expert
  - 4b2 The knowledge modelling expert analyses the case and proposes a solution
  - 4b3 Return to step 3

## 5.2 UC2: Evolution: Register request for content change

**Trigger:** A team steering committee meeting takes place

**Preconditions:** A case is on the meeting agenda

**Success guarantee:** The case is rejected or approved for implementation

**Main success scenario:**

1. Any time between when the case is proposed for discussion and the meeting, committee members may assess open cases and add business-, technical- or implementation-related comments.
2. During the meeting, the asset manager presents the case.
3. The steering committee members discuss and comment on the case

4. The steering committee approves the case for implementation
5. The asset manager schedules the case for implementation

**Extensions:**

- 4a The case is rejected:
  - 4a1 The steering committee rejects the case along with a justification
  - 4a2 The request manager informs the client and provides recommendations
- 4b The case needs additional input:
  - 4b1 The steering committee rejects the case along with a request for action, information or agreement for an alternative proposal
  - 4b2 The request manager informs the user and requests additional actions, information or agreement for an alternative proposal
  - 4b3 The request manager registers a request for content change (UC1)

## 5.3 UC3: Implementation: Implement request for content change

**Trigger:** A case is scheduled for implementation

**Preconditions:** The case is approved for implementation

**Success guarantee:** The case is implemented and validated, while the data is exported and stored in a common repository

**Main success scenario:**

1. The request manager schedules a case for implementation
2. The data authoring officer reads the case and executes the content authoring accordingly
3. The data authoring officer automatically, or assisted by the data processing officer:
  - (a) exports the asset from the authoring tool, and

- (b) runs the SHACL validation for conceptual and structural issues, and
  - (c) runs the difference calculation between exported content and the previous release export, and
  - (d) runs the fingerprint calculation for the exported content
  - (e) the data and reports are stored in the common repository
4. The data authoring officer checks:
- (a) (verification) that the diff report calculated between the previous release export corresponds with the implemented change request<sup>1</sup>.
  - (b) (validation) that no structural anomalies are present in the fingerprint and validation reports.
5. Repeat steps 1 - 4 until all cases are implemented for the asset
6. The data authoring officer informs the quality assurance officer of the successful implementation of all cases.

**Extensions:**

- 2a Translations are necessary:
- 2a1 Additionally, the data authoring officer manages the necessary translations and proof-reading (the process is described elsewhere: export selected data for translators, send to the translation unit, import updated data containing the translations, validate and proofread the translations)
- 4a Implementation or data is invalid:
- 4a1 The data authoring officer collects and documents all issues
  - 4a2 Return to step 2

---

<sup>1</sup>This is to validate that the export reflects change request case for the change request ticket, keeping the editors on the safe side. If all is good, then this is the final diff.

## 5.4 UC4: Validation: Validate the request for change

**Trigger:** An implementation is scheduled for validation (data available in SRC-AP [12] format along with assessment reports)

**Preconditions:** All cases are implemented for an asset and no further updates are foreseen

**Success guarantee:** The data is validated by a second “pair of eyes” and is marked as fit for publication

**Main success scenario:**

1. The data authoring officer provides the data and validation reports in the common repository
2. The quality assurance officer verifies that the diff report corresponds to the case requirements
3. The quality assurance officer checks the fingerprint and validation reports for semantic or structural anomalies
4. The quality assurance officer accepts the implementation and the data and informs the asset manager
5. The asset manager marks the asset as fit for publication.

**Extensions:**

- 4a Data quality issues are detected:
  - 4a1 The quality assurance officer identifies and documents issues in the validation and fingerprint reports and informs the data authoring officer what the issues are and eventually explains how to fix them.
  - 4a2 The data authoring officer implements the request for change (UC3)
- 4b Implementation issues are detected:
  - 4b1 The quality assurance officer identifies and documents issues in the diff report and informs the data authoring officer what the issues are and explains how to fix them.

4b2 The data authoring officer implements the request for change (UC3)

## 5.5 UC5: Release: Prepare the publication content

**Trigger:** Asset release is requested

**Preconditions:**

- The data is conceptually and formally validated (SRC-AP) and its content is fit to be published
- A code freeze is declared; no more changes are foreseen

**Success guarantee:** The data is available in standard (and, where requested, additional) forms and formats.

**Main success scenario:**

1. The asset manager requests a release
2. The data processing officer starts the transformation processes from SRC-AP form into:
  - (a) Target forms: SKOS-AP-EU [14], SKOS-AP-EU-Act, SKOS-core [36]
  - (b) Target formats: RDF/XML [6] , Turtle [10] , JSON-LD [46]
3. The data processing officer runs the fingerprinting and SHACL validation for structural issues and confirms the transformation went well<sup>2</sup>.
4. The data processing officer start the transformation processes from SRC-AP/SKOS-AP-EU into required additional forms and formats such as CAT-XML, XSD, Genericcode, Excel/CSV, MarcXML, GeoJSON, etc.
5. The data processing officer places the generated output into the common repository along with the validation reports, and informs the asset manager and the publication officer

**Extensions:**

---

<sup>2</sup>This process is automatic and has the purpose of ensuring the transformation process passed correctly, keeping the data processing officers on the safe side.

- 3a The validation reports reveal content related issues:
  - 3a1 The data processing officer identifies and documents the issues and reports them to the quality assessment officer
  - 3a2 The data authoring officer implements the request for change (UC3)
- 3b The validation reports reveal data-related issues:
  - 3b1 The data processing officer identifies and documents the issues and informs the quality assessment officer
  - 3b2 The data processing officer fixes the issues due to the transformation process
  - 3b3 Return to step 2

## 5.6 UC6: Publish: Publish a reference data asset

**Trigger:** A publication of selected assets is requested

**Preconditions:**

1. The selected assets, validated and converted into all the necessary forms and formats, are available in the common repository
2. The asset user manual is available in the common repository
3. Format user manuals are available in the common repository
4. Asset metadata, both content-related and technical, are available in the common repository

*Success guarantee:* The updated assets are accessible on selected dissemination platforms and the broad public is informed about the new publication

*Main success scenario:*

1. The scheduled publication due date occurs
2. The publication officer generates the release notes from the diff report that summarises what has changed (in more detail than the impact assessment).

3. The publication officer generates the publication summary and impact assessment report (having sections customised for each major stakeholder) that presents an overview of the main content changes and if structural changes are included.
4. The asset manager checks the release notes and the impact assessment (to ensure that they reflect the change request cases)
5. The request manager sends the publication summary and impact assessment reports to the stakeholders, to inform them of upcoming changes and to collect any pre-publication feedback.
6. The publication officer runs the packaging process for each asset (parallel to the impact assessment process) resulting in the generation of:
  - (a) Additional technical metadata (DCAT [54], METS [13], IMMC<sup>3</sup> , etc.)
  - (b) Packages (ZIP or other) for selected dissemination platforms (Cellar [22], ODP, Bartoc [32], Joinup [26], Wikidata [52], etc.) that contain all the necessary content, documentation and metadata
7. The publication officer tests the integrity/fitness of the generated packages by using the validation mechanisms offered by the dissemination platforms (validators or test dissemination environments)
8. The request manager receives implicit acceptance of the impact assessment from stakeholders (i.e. no objections were raised within the established deadline) and informs the publication officer that the assets can be uploaded to the dissemination platform(s).
9. The publication officer publishes the packages to the dissemination platform, tests that the assets are accessible and informs the asset manager that publication has been successfully completed.
10. The request manager informs the broad public (including stakeholders) that the publication has been completed.

---

<sup>3</sup>see <https://op.europa.eu/en/web/eu-vocabularies/immc>

## 5.7 UC7: Publish: Publish a model asset

**Trigger:** A publication of selected assets is requested

**Preconditions:**

- The selected assets, which were approved and converted into the standard forms and formats, are available in the common repository
- The asset user manual is available in the common repository
- Format/representation user manuals are available in the common repository
- Asset metadata, both content- and technical-related are available in the common repository

**Success guarantee:** The assets are accessible to the broad public on the selected dissemination platforms

**Main success scenario:**

1. The scheduled publication due date occurs
2. The publication officer automatically generates the release notes which summarise the content of the publication.
3. The request manager sends the publication summary to inform them of upcoming changes and collect any pre-publication feedback.
4. The publication officer runs the packaging process for each asset (parallel to the impact assessment process), resulting in the generation of:
  - (a) Additional technical metadata (DCAT, METS, IMMC, etc.)
  - (b) Packages (ZIP or other) for selected dissemination platforms (Cellar, IMMC, ODP, Wikidata, Bartoc, Joinup, etc.) that contain all the necessary content, documentation and metadata
5. The publication officer tests the integrity/fitness of the generated packages by using validation mechanisms offered by the dissemination platforms (validators or test dissemination environments)
6. The publication officer publishes the packages to the dissemination platform,



tests that the assets are accessible and informs the asset manager that publication has been successfully completed

7. The request manager informs the broad public (including stakeholders) that the publication has been successfully completed

**Extensions:**

6a Packages are rejected by the dissemination system:

- 6a1 The publication officer contacts the support team of the dissemination system and resolves the issue.
- 6a2 In case the generated package is incorrect, the publication officer corrects the generation processes.

# Chapter 6

## Business architecture

This chapter covers the business architecture. The focus falls almost entirely on the bottom layer of the business architecture structure (see Figure 6.1) describing the internal processes, events and roles answering questions concerning who shall do what and when.

We address here both the current organisation and the new organisation of the asset life-cycle process.

First we establish a baseline representing the current setup and, second, we present how the new processes will look like in the light of digital transformations moving towards goals identified in the motivation structure (Section 4).

Then we explain the general idea of how the business architecture is structured, and which serves as an interpretation framework for the following diagrams.

### 6.1 Prototypical business structure

Following the metaphor of layers presented in the motivation view, the organisation of business structure is also explained in terms of layers. Figure 6.1 depicts three layers with the most important elements of the business structure.

The topmost layer accounts for the external players or *actors*, which represent a business entity that is capable of performing behaviour and *roles*, which represent

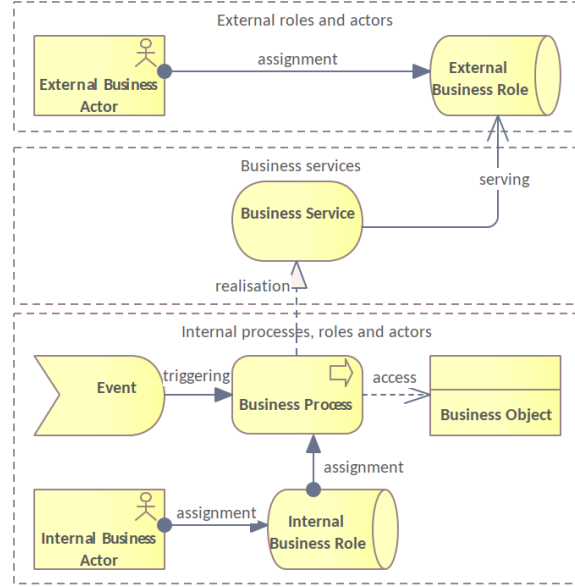


Figure 6.1: The prototypical business structure view

skills and responsibilities for performing specific behaviour, and to which an actor can be assigned [50].

The middle layer represents the *services* that are offered by the organisation to external players. A business service represents explicitly-defined behaviour that a business role, business actor or business collaboration exposes to its environment [50].

The lower layers accounts for the internal organisation in terms of *events*, *roles*, *processes* and *objects*. The business process represents a sequence of business behaviours that achieves a specific result such as a defined set of products or business services. The business event represents an organisational state change; while a business object represents a (passive) concept used within a particular business domain.

## 6.2 Current asset life-cycle stages

The current asset life-cycle process is organised in six stages: *inception (or evolution)*, *implementation*, *pre-release*, *release*, *publication* and *consumption*. Each of the stages represents a business sub-process. Figure 6.2 depicts the order in which

stages flow and indicates that each stage process accesses a data asset, the central artefact in the diagram.

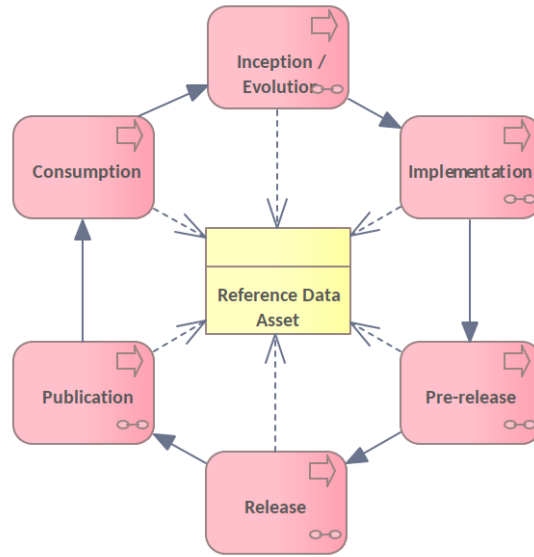


Figure 6.2: The current asset lifecycle stages

The *inception* stage means that a request arrives for creating and publishing a new data asset and it is being dealt with by the team. The *evolution* stage is similar, only that the request is one for change of an existing data asset. There is no difference in the way these two requests are treated and so the stage name is a conflation of the two. This stage also includes negotiating the request back with the client and then finally deciding and planning its implementation and publication.

The *implementation* stage deals with actually changing, authoring, converting (MS Excel to XML and back) and verifying the client request.

*Pre-release* marks that the content has been implemented accordingly and can be validated by a second pair of eyes implementing the *four eyes principle* implemented in SU. This verification and validation is performed by checking the validation reports and by comparing the difference between current and previous versions of the asset conveyed in a diff report.

In the *release* stage the validated content is placed in a dedicated location of the common repository which indicates that the content is fit for publication.

The *publication* stage deals with packaging the content and disseminating it to the selected data disseminators, Cellar being the most important. During this stage, a set of announcements and communications ensure that the main stakeholders and the broad public are aware of the published new version of the asset.

*Consumption* stage is the one that happens outside SU borders. It is clients who use the data and then, in the process, come up with additional requests for either changing existent assets or adding and publishing new ones.

## 6.3 Actors and roles

This section describes identified actors and roles relevant to the asset life-cycle process. Figure 6.3 depicts their relations.

*Asset manager* (a mix of *operational and business data steward*) is primarily responsible for data content, context and associated business rules. This role is characterised by full responsibility for the asset quality, enforcing policies and data governance processes, and ensuring asset fitness (both content and metadata) to business needs. In the SU, this role is also responsible for high-level interaction with main stakeholders and important clients.

*Team steering committee* (also known as the team meetings) is a body composed of business, technical and analytical roles whose main purpose is to provide executive and operational guidance validating business requests and assessing both data management and broader impact, determining implementation priority, as well as promoting data governance and standardisation practices.

*Request manager* is the interface with the client collecting change requests, assessing business needs and translating them into data management requirements, all being summarised and documented case-by-case. *Data authoring officer* is responsible for editing data in a content management system implementing the cases prepared by the request manager. Quality assurance officers validate that the content implementation is correct from both technical and business points of view.

*Data processing officer* is a technical role that is responsible for preparing the assets for publication. The responsibilities include, but are not limited to, data storage, manipulation, automatic transformation and generation of validation and assessment reports.

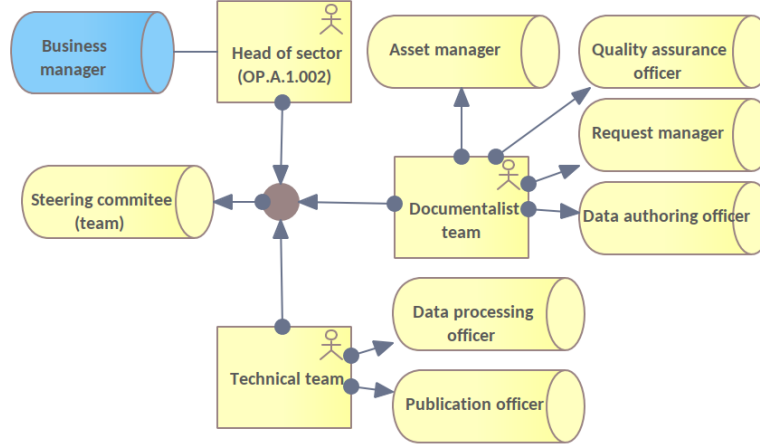


Figure 6.3: The internal roles in metadata and reference data sector

*Publication officer* is a technical role responsible for packaging and disseminating assets to specialised platforms.

*Stakeholder steering committee* is a body representing the main clients and stakeholders ensuring data and model harmonisation, alignment of data management practices and adoption of international standards.

*Client* (*change requester* and *data user*) is a generic external role who, on one side, consumes data and services provided by the Standardisation Unit and, on the other side, suggests publication of new assets or modification of existing ones.

*Data disseminator* is an external role providing the Standardisation Unit with reliable data dissemination capabilities which are meant to make assets available for clients.

The external roles and stakeholders have already been addressed in the motivations structure depicted in Figure 4.2. Each of these roles has a corresponding element in the business model and will be employed accordingly.

## 6.4 Data stewards and custodians

There are two roles that are not used in this architecture but are of high relevance in this context. In Data Governance groups, responsibilities for data management are

increasingly divided between the business process owners and information technology (IT) departments. Two functional titles commonly used for these roles are Data Steward and Data Custodian.

Data Stewards are commonly responsible for data content, context, and associated business rules. Data Custodians are responsible for the safe custody, transport, storage of the data and implementation of business rules. Simply put, Data Stewards are responsible for what is stored in a data field, while Data Custodians are responsible for the technical environment and database structure.

The data custodians are the agents that ensure that the content is coherent and complete. It is also responsible for the content correctness and harmonisation among multiple stakeholders and its usefulness in broader context of application.

These two roles are not employed neither in the current nor in the new architecture. They are, however, described here in order to encourage the SU towards adoption of these roles in the future.

## 6.5 Current asset life-cycle overview

This section assembles the asset life-cycle process stages and the main internal roles together into an overview diagram depicted in Figure 6.4. It indicates what roles are assigned to which processes, along with a cyclical depiction of the process sequence. Next we comment on the involvement of each role in the asset life-cycle process. All the life-cycle stages are internal to the SU except for the last one, consumption, which takes place at the client's premises.

In the inception/evolution stage, the request manager is responsible for creating, documenting and ensuring descriptive completeness for requests arriving from clients to change existing assess or to create new ones. These requests are managed as individual or, on some occasions, interdependent cases. This role serves as the primary interface with third parties. For this reason, in the publication stage, this role communicates with stakeholders and broad public about asset changes when they are published.

The asset manager is in charge of analysing and summarising the request case in the inception/evolution stage. This role intervenes in the pre-release stage to acknowledge that the case has been implemented and the asset is fit for publication; then,

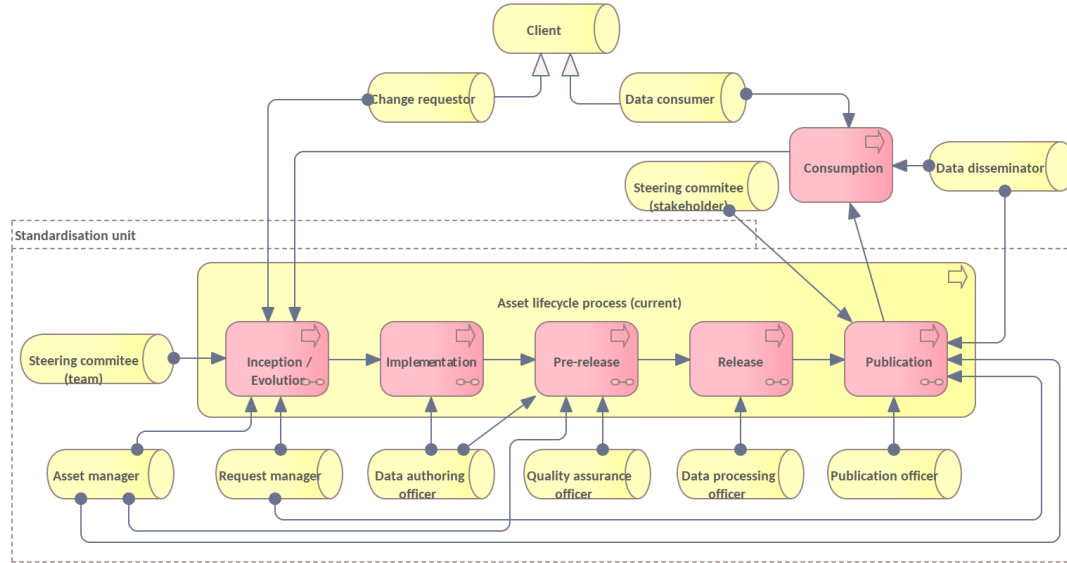


Figure 6.4: The current asset lifecycle stages and roles

in the publication stage, to check the impact assessment and release notes before they are used in the communication with external parties.

The team steering committee is involved in the initial stage only. After a new change request case is created, the team steering committee decides whether the case shall be further processed and, if so, then the decision is about when and how.

The data authoring officer is responsible for the case implementation and, in the pre-release stage, for verifying and validating their own work as the “first pair of eyes” (of the “two pairs of eyes” principle). The “second pair of eyes” verifying and validating the case implementation is enacted by the quality assurance officer in the same pre-release stage.

Once the case is marked as fit for publication, in the release stage, it is placed automatically, or by intervention of the data processing officer, in a region of the common repository tagged for “release”. If any technical issues are encountered, then the data processing officer intervenes and fixes them.

In the publication stage, the publication officer generates the release artefacts and release notes, packages the assets and disseminates them to the dissemination partners. External steering committees, such as IMMC metadata sub-committee, GIL



EuroVoc and others are asked for final feedback two weeks in advance before final dissemination.

Next, we address the asset life-cycle stages in more detail as elicited from the technical and business teams of the SU. These descriptions are not covering ultimate details of the process, but aim at describing the important building blocks of the current stages.

## 6.6 Current inception and evolution stage

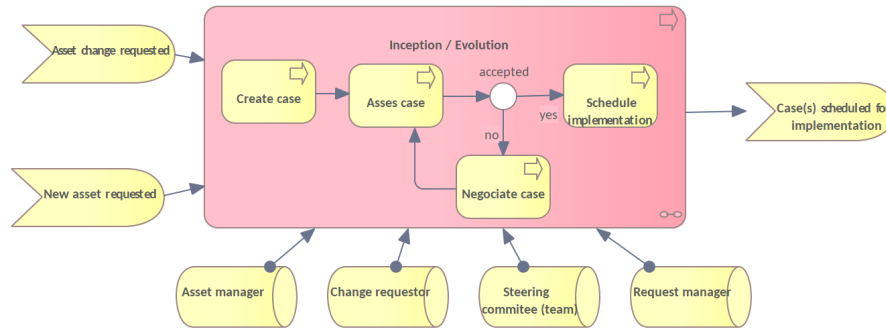


Figure 6.5: The current process for the inception and evolution stage

The process, depicted in Figure 6.5, starts when a client request arrives to either update a data asset or create a new one. This case, after being recorded, analysed and summarised, is assessed by the asset manager and then discussed in the team meeting.

When the case is accepted, then it is scheduled for implementation. Otherwise, i.e. when it is not (initially) accepted, the case enters a “negotiation” stage, due to one of two things: either the case is incomplete and more information is required from the client, or the case is unacceptable and a rejection is provided to the client with an explanation of why or possibly even with suggestions / counter-proposals.

Client communication is mainly carried out by email or by telephone, with the cases being managed using the Jira ticket management system [5].

The process ends when either the case is rejected or when it is scheduled for implementation.

## 6.7 Current implementation stage

Figure 6.6 depicts the current implementation process. It starts when the case is queued for implementation. The data authoring officer modifies the asset content according to the instructions provided in the case description. The editing takes place in an MS Excel [34] workbook which represents an interface to the asset content. MS Excel is the main editing tool. Once the changes are complete, the workbook is committed into the SVN repository [2], which triggers an automatic conversion of the MS Excel workbook into an XML form [37]. The XML form is considered the primary asset source structured with CAT-XSD scheme, and for this reason, sometimes we call it “CAT-XML”. It is further converted back into MS Excel workbook form, this way entering a conversion loop which also serves as validation mechanism ( $XML \rightarrow Excel \rightarrow XML \rightarrow Excel$ ).

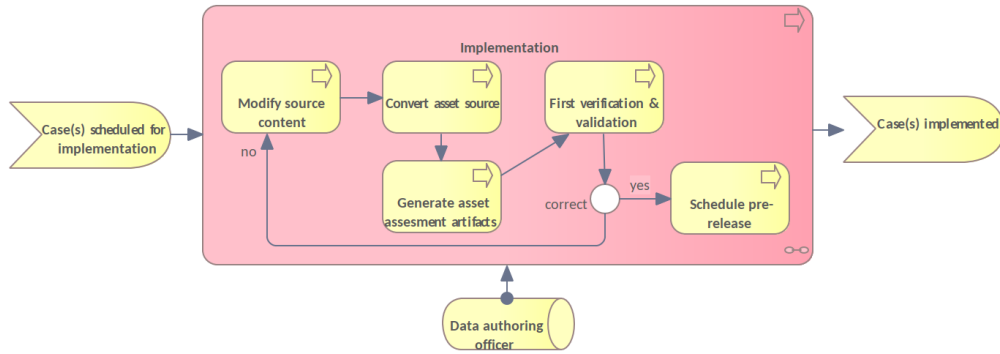


Figure 6.6: The current process for the implementation stage

Once the asset is converted into XML form, it becomes possible through a set of Perl scripts and XSLT style-sheets [27] to automatically generate assessment artefacts such as the diff report and the schema validation report. The diff report indicates what changes have been done to the content between the previous and the most recent versions, while the validation report details violations, if any, of XML structural constraints.

The editor then verifies the diff report to ensure the case implementation completeness and that the asset can be tagged for pre-release. Otherwise, the content should be edited once again.

The process ends with the asset being marked for pre-release, which means that the

case implementation is complete.

## 6.8 Current pre-release stage

The pre-release stage is depicted in Figure 6.7. Once the case is marked “implemented”, and the assessment artefacts were generated after the conversion into XML form, the second verification and validation can be performed by the quality assurance officer.

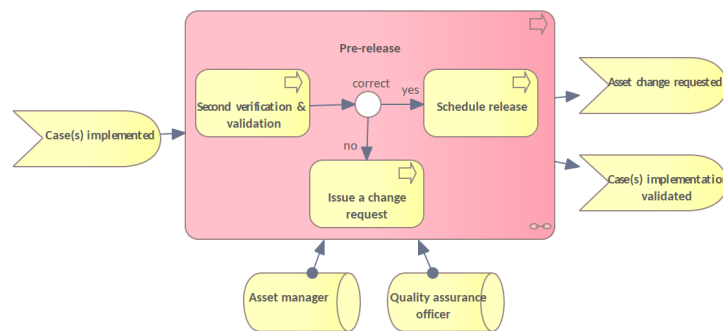


Figure 6.7: The current process for the pre-release stage

If issues are identified in the case implementation, the quality assurance officer creates a new request for changes and the case returns to the implementation stage; otherwise, the case implementation is validated and the asset is marked as ready for release.

This process involves mostly manual steps. Some minor content transformation can take place such as improving the serialisation of RDF files (called internally as RDF “prettification”), as well as an additional validation of record identifiers in the XML file. These operations, however, are merely technicalities and do not have any business relevance. Therefore they are omitted in the process diagram from Figure 6.7.

## 6.9 Current release stage

The release stage is a symbolic step after an asset has been verified and validated by four “pairs of eyes” and it has been confirmed that all cases have been correctly

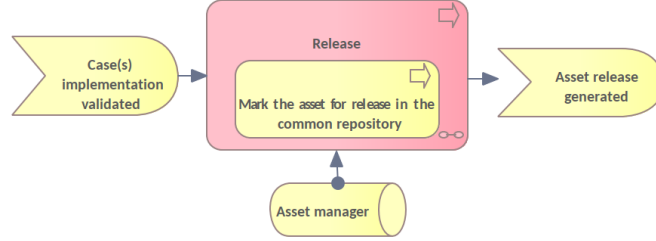


Figure 6.8: The current process for the release stage

implemented, also that the asset is fit for release in the subsequent publication. This is depicted in Figure 6.8.

The release stage is realised by copying the updated version of the asset into a special area of the common repository and marked with the “release” tag.

## 6.10 Current publication stage

The current publication stage is depicted in Figure 6.9. It is a wide-ranging process that involves almost all the various roles.

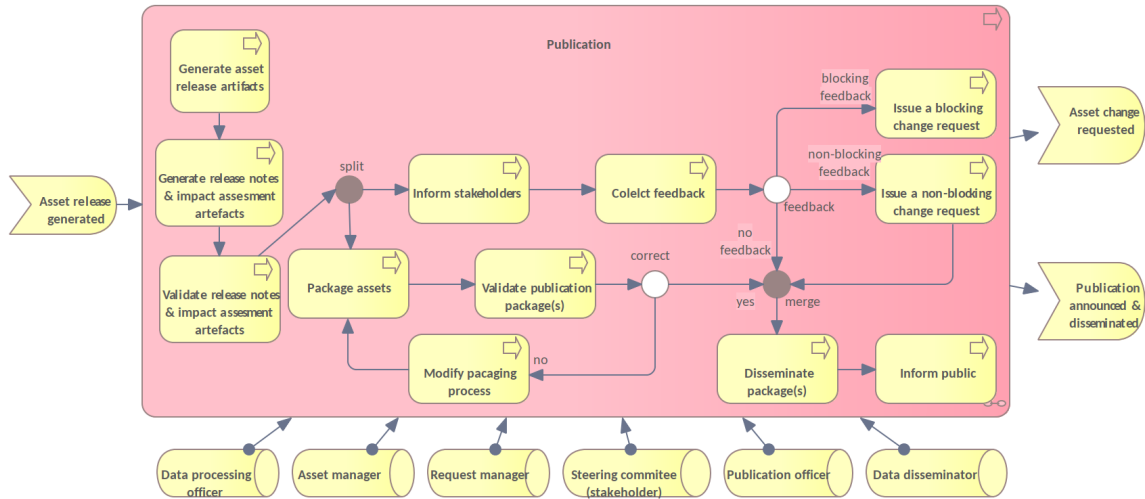


Figure 6.9: The current process for the publication stage

The publication process starts several weeks before the scheduled publication date,

when a code freeze is announced and all pre-selected assets are marked for release. It commences by generating (from the CAT-XML source) a selection of forms and formats to ease asset consumption by various clients. Mostly XSD, CAT-XML and SKOS-AP-EU forms are generated that are serialised in XML, Turtle and JSON formats. In addition, some assets are also prepared in Genericcode, MS Excel/CSV, MarcXML, GeoJSON and other formats.

Next, the publication notes are prepared together with a few different impact assessments specially prepared for targeted stakeholders. These communication artefacts are checked for correctness and sent to the corresponding stakeholders. After a predefined number of days (usually two weeks) the feedback, if any, is collected. Reception of no feedback is considered as a tacit acceptance of the current publication and that the process can continue. Seldom is blocking / non-blocking feedback received which leads to creation of new change requests and, depending on the situation, last-minute changes are executed. On the other hand, if the feedback received is non-blocking, the intervention is scheduled for the next publication.

In parallel, a packaging process is executed during which the assets are prepared for dissemination. They are assembled in packages accompanied by their content-related and technical metadata, user manuals, format documentation and, of course, the asset itself expressed in all pre-generated forms and formats, the release artefacts. The packages are verified to establish if they can be accepted by the target dissemination platforms.

Finally, the packages are disseminated and the successful publication is announced to the broad public. The dissemination is done primarily through Cellar[22], although other dissemination channels are also employed, among which are Wikidata[52], Publications Office Open Data Portal (ODP), Bartoc [32], JoinUp platform [26].

This section completes the detailed presentation of the asset life-cycle process as it is today.

The digital transformation currently undertaken by SU management, that is in part targeted by this architecture, has an impact on the asset life-cycle business, application and technical architectures. The next sections describe how the new asset life-cycle architecture is envisaged.

## 6.11 New asset life-cycle overview

The structure of the new asset life-cycle process is depicted in Figure 6.10. In this new architecture, we propose incremental changes, so as to cause minimal disruption to the team and ongoing operations.

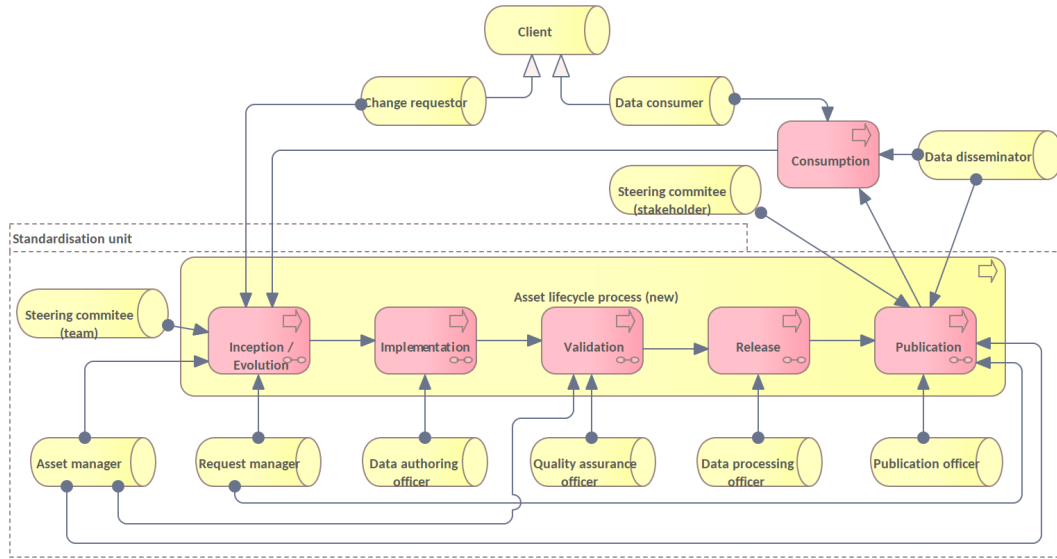


Figure 6.10: The current asset life-cycle stages and roles

This process is similar, in structure and stage names, to the current one: i.e. six stages are circularly linked. The only difference is the replacement of the pre-release stage by a validation stage. There are, however, significant changes in the structure of three stages: implementation, validation and release, while the inception/evolution stage is identical to the current one, and the publication stage is almost entirely unchanged. These changes are addressed in detail in the following sections.

There are no new roles in the new process; however, there is a difference in the role allocation and involvement (of the various roles) at different stages of the process compared to the current one.

As mentioned above, the implementation/evolution stage is identical to the current one. So, after the case has been registered and processed, the implementation is scheduled. How it happens in the new workflow is presented in the next section.

## 6.12 New implementation stage

The new implementation process is depicted in Figure 6.12. The main digital transformation at this stage is adoption of the RDF asset source, and the switch from MS Excel as the content editor to VocBench3 [47, 48] semantic web editor.

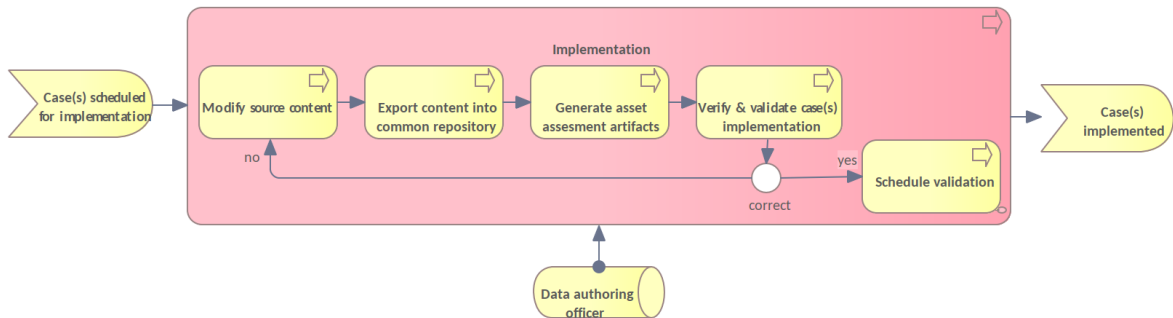


Figure 6.11: The new process for the implementation stage

The overall process looks very similar to the current one; however, the employed technology for the source editing, i.e. VocBench3 [48] and SKOS model(s) [36], render an entirely different editing experience when modifying the source content to implement the request case.

When the editing is complete, the content is exported from VocBench into the common repository and a set of asset assessment artefacts are generated. These artefacts include the asset diff report, the fingerprint report and the validation report (based on SHACL [30] validation rules). These reports are no longer implemented based on XSLT technology [27], because the underlying source is the new RDF representation [55] and no longer (CAT-)XML. More technical details are addressed in the application architecture in Section 7.4.

The editor (data authoring officer) then verifies that the case is implemented correctly and, if so, then validation is scheduled to be undertaken by the quality assurance office. Otherwise, if some issues are detected, then more editing is performed in VocBench3.

## 6.13 New validation stage

The validation stage is new in the asset life-cycle process. Here, the correctness assessment is separated into two steps: verification and validation. This is a continuation of the current conception of how the case implementation is being assessed. The verification primarily focuses on the business aspects of the case implementation, whereas the validation deals with technical and more formal aspects of the content structure and consistency. As the new technology allows for semantic accounts, then validation also extends to deal with semantics.

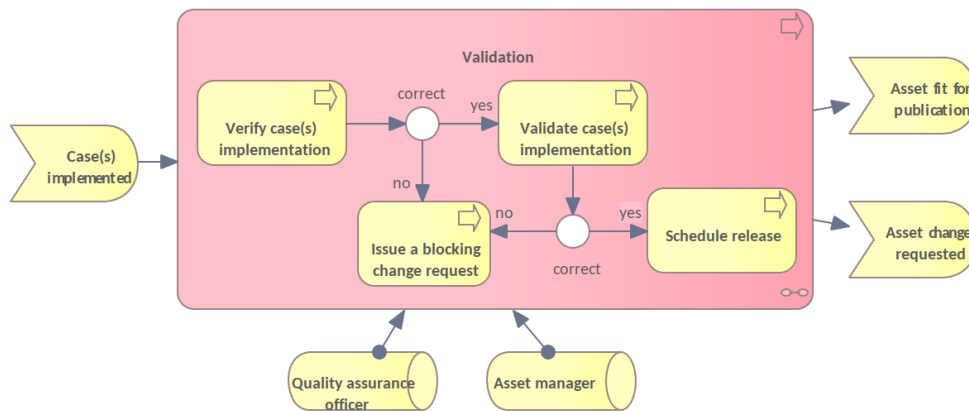


Figure 6.12: The new process for the validation stage

In case errors are detected in the implementation, the quality assurance officer issues a change request and the case goes back into the implementation stage. Otherwise, the case implementation is considered correct and, from then on, a release can be scheduled. This is performed by the asset manager.

## 6.14 New release stage

The new release stage, depicted in Figure 6.13, differs considerably from the current one. This stage is no longer one of simply marking the asset implementation as “fit for publication”, but deals with all data transformations and preparation of artefacts necessary in the publication stage.

The release starts when all change request cases, in the scope of the scheduled publication, have been implemented and validated. Then, the primary release artefacts



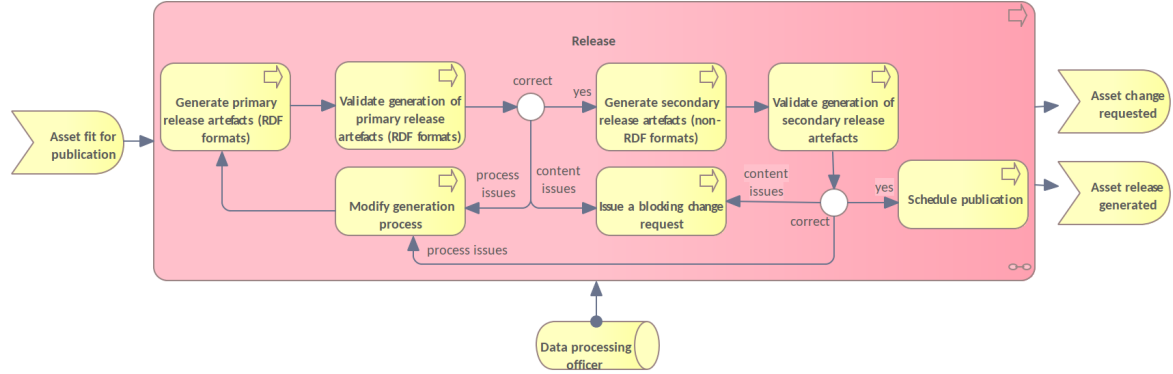


Figure 6.13: The new process for the release stage

are generated from the RDF source is expressed in SRC-AP form [12]. The main artefacts are the source in SKOS-AP-EU form [14], a special extension of the SKOS-AP-EU which is necessary for publishing the content in Cellar because of backward compatibility issues (called SKOS-AP-EU-Act), and the SKOS core [36] representation. These artefacts are also generated in RDF/XML [42, 6], Turtle [10] and JSON-LD [45, 46] formats.

The transformation process is accompanied by a set of validation processes meant to safeguard the output and ensure that the transformations are executed correctly. The validation reports are checked by the data processing officer. If process-related issues are detected, then most likely the processes contain bugs and need to be fixed. If content related issues are spotted, then a new change request is created and the process reverts to the implementation stage.

After primary release artefacts are created, then secondary release artefacts are created. All of them are non-RDF forms that are currently produced for clients (CAT-XML, XSD, Genericcode, MS Excel/CSV, MarcXML, GeoJSON, etc.) and must continue so. An automated validation of the generated assets, to the greatest extent possible, secures and ensures quality of the output. As in the case of primary artefacts, if errors are detected then depending on their nature, either the data transformation process must be updated or a content change request is issued and the process reverts to the implementation stage.

The transformation technology employed here needs to provide ETL<sup>1</sup>/ELT<sup>2</sup>-like capabilities for both RDF and data formats. An extended discussion about application capabilities is covered in Section 7.6.

At the end of this stage, all important asset transformations and data conversions must be complete and the necessary forms and formats consumed by target clients shall be available for publication. Finally, assets are copied into the special place within the common repository available to the publication process.

## 6.15 New publication stage

The new publication stage process is depicted in Figure 6.14. This process is very similar to the one currently performed and that is described in Section 6.10.

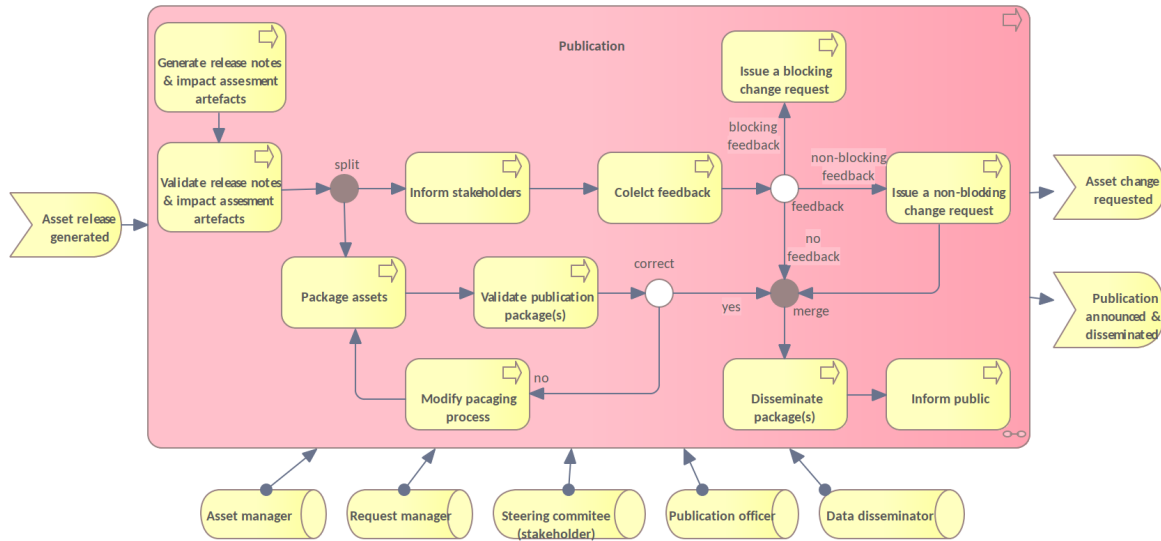


Figure 6.14: The new process for the release stage

The main difference in this process is the starting point. If currently it starts by transforming and converting the asset source into all forms and formats demanded

<sup>1</sup>Extract, transform, load (ETL) is the general procedure of copying data from one or more sources into a destination system which represents the data differently from the source(s) or in a different context than the source(s).

<sup>2</sup>Extract, load, transform (ELT) is a variant of ETL where the extracted data is loaded into the target system first.

by clients, then the new process starts from the premise that the entire publication content has already been generated.

The new process starts by the generation of release notes and set of impact assessment reports destined for different stakeholders. These communication artefacts are verified by the asset manager and then the request manager disseminates the news to the stakeholders and the broad public about the upcoming publication. The stakeholders may have a veto or comment on the current publication, just as is currently foreseen.

What is left, at this stage, from the technical point of view, is packaging and distributing the publication packages to the data disseminators. The process ends when the latest version of published assets are accessible via the selected data disseminators and the broad public is informed.

This section finishes the description of the business architecture. Next is presented the application architecture accounting for necessary services and capabilities that must be in place to serve the current business processes along with an account of what components realise such services.

# Chapter 7

## Application architecture

This section covers the application architecture (term from ArchiMate language). We present here the essential services and the components realising those services to enact the current and the new life-cycle processes.

### 7.1 Prototypical application structure

This section presents the application architecture from the solution architecture point of view. A generic solution architecture is depicted in Figure 7.1.

The application architecture presented covers the application as a “white box”, its internal component structure, services and interfaces with adjacent applications. Typically the solutions architecture takes the technology aspects into account, accounting for parts of the infrastructure.

The central element of the application architecture is the *application service*, which represents application behaviour or functionality. The application services, from an inter-layer perspective, serve the processes in the business layer and provide support for their realisation.

The application services are realised through application processes. The processes have application components assigned to them signifying their place of encapsulation. Application components are modular and replaceable blocks encapsulating implementation of application services and functionalities. In practice, for clarity, we

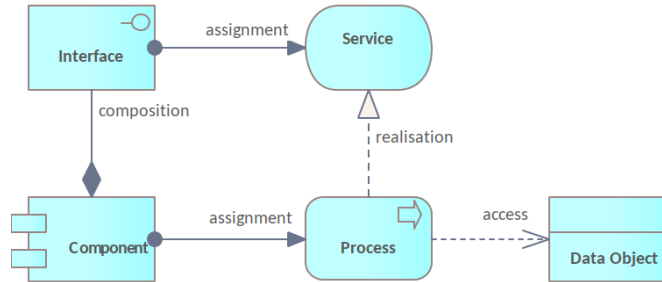


Figure 7.1: The prototypical application structure view

take a shortcut, and say that the application services are realised through *application components* directly.

Components are said to expose interaction *interfaces* which are modelled, in ArchiMate, as proper parts of the components. The interfaces are assigned to services signifying how the latter are to be accessed and consumed.

Also, components, as well as processes they encapsulate, access *data objects*, which are passive components of the application architecture.

The solution architecture presented in this section is an adaptation of the generic architecture. Here we focus on presenting what application services are used to support each business process. Moreover, we are interested in grasping the difference in the application layer, between the current and new versions of the business processes.

To do so, we split the application view diagrams into three vertical lanes. The left lane hosts the current version of the business process as well as the application services and components that are used to support it. In the right lane, we place the new business process and the new application services and components that will have to be adopted for the digital transformation. The middle lane hosts the services and components that are currently employed and will be carried over into the new application architecture: they are common to both the current and new architectures.

Below we present an overview of the application architecture, in terms of services alone, depicting how the asset life-cycle stages are served.

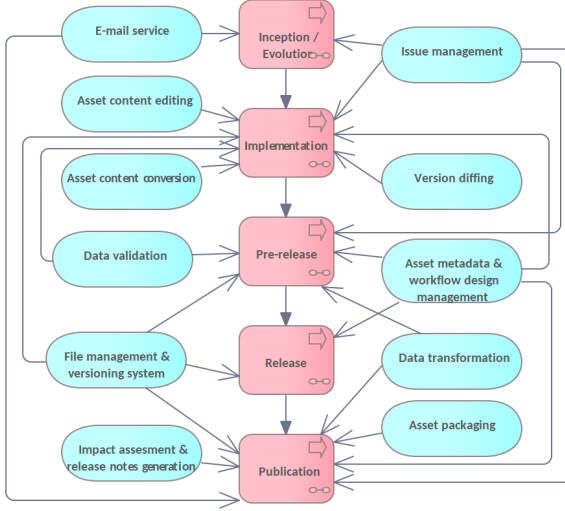


Figure 7.2: The application services that serve the current asset lifecycle

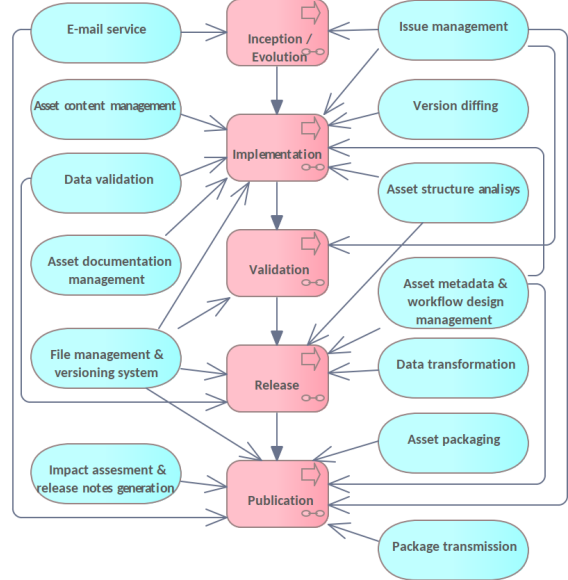


Figure 7.3: The application services that serve the new asset lifecycle

## 7.2 Current and new application service architecture

This section presents how the asset life-cycle stages are supported by the application services. The current application architecture is depicted in Figure 7.2 and the new one in Figure 7.3. The services employed in each of the architectures are mostly the same, with a few exceptions. What differs more significantly is their utilisation in the asset life-cycle stages. For this reason, we set them side-by-side to provide a contrasting image.

We start the description from the top of the diagram following the sequence of process stages. First is the *e-mail service*. It represents the entire set of capabilities for sending emails covering both the email server and email client. This is realised via the Outlook system [33].

The *issue management* service represents the capability of recording, documenting and analysing a change request case in a distributed collaborative manner between multiple roles and actors. This is realised via the Jira system [5].

*Asset content editing* is the service which enables the documentalists in the team to modify asset content and hence implement the request cases. This service is realised by MS Excel desktop software [34].

The *asset content conversion* service implements the conversion between MS Excel workbook and CAT-XML forms of the asset content. The two representations are equally expressive and the conversion process runs in both directions  $Excel \rightarrow XML$  and  $XML \rightarrow Excel$ .

*Version diffing* is a service which calculates and presents the difference between two versions of the same asset. The *data validation* service is self-explanatory. It checks whether the asset content is correct, or not. The asset content conversion, version diffing and data validation services are realised by components that comprise the legacy workflow, which is currently in use.

*Asset document management* provides capabilities for describing, storing and producing documents about assets in various human-readable formats (e.g. HTML, PDF, docx). It also allows for document metadata management. Currently, this capability is realised by PDF forms, called Asses Document Description (ADD), which are produced from XML-DITA sources [15, 16] using Adobe FrameMaker [1].

*Asset metadata & workflow management* is a service which centralises the flow of the automated processes. It is currently realised by legacy workflow implementation.

The *file management & versioning* service enables storing the content and and tracing the evolution of assets. This service is realised by the SVN system [2].

The *data transformation* service is self-explanatory. It stands for a generic capability of transforming data from one form and format into another. Currently, this service is realised by a multitude of custom-built transformation procedures, some of which are chained to produce the desired outcome. This service is realised by components that are part of the legacy system.

The *asset packaging* service prepares assets for transmission to the partner distribution systems, specifically Cellar. It covers functionalities including assembling the necessary asset representations together with their documentation, generating necessary technical metadata and zipping the assets in a manner that is acceptable for partner distribution systems. The type of package and the asset metadata descrip-

tion varies, yet among the most prominent ones are METS [13], IMMC<sup>1</sup> and DCAT [54].

The last service to mention that is employed in the current life-cycle process is the *impact assessment & release note generation* service. Its title is again self-explanatory. The values it provides are the release notes that are published with assets and describe the list of changes implemented in each version. The impact assessment is a type of report which includes assessments for target stakeholders which usually are service providers whose systems depend on the assets published by the SU. These special impact assessments may target whether a given aspect of an asset has changed which may disrupt functioning systems. So, a series of specific checkers and reports are produced to safeguard the partners.

The new application architecture (Figure 7.3), as mentioned above, re-configures how services are used across process stages. It replaces the asset content editing service by a new one which is *asset content management*. This service is realised by a fully-fledged semantic web editing system – VocBench3 [48]. In addition, two more services are added: asset structure analysis and package transmission services.

The *asset structure analysis* implements an automatic fingerprinting of the asset structure which provides an insight into how the managed asset is realised and substantiated. This fingerprinting serves as an indicator for the structural validation in the implementation step. For example, a missing or extra property will be spotted with this service.

The *package transmission* provides the possibility to automatically deliver and ingest the prepared packages into the dissemination system, rather than the publication officer having to perform this operation manually.

## 7.3 Inception and evolution services and components

In the first stage of the asset life-cycle, the technical requirements are limited to client communication and the request documentation services as depicted in Figure 7.4.

In the left and right lanes of the diagram are where the current and new life-cycle

---

<sup>1</sup>see <https://op.europa.eu/en/web/eu-vocabularies/immc>



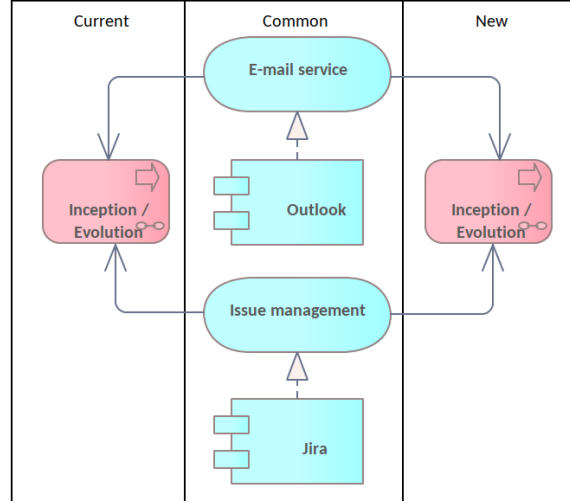


Figure 7.4: The application services and components that serve the current and new inception and evolution stage

stages are represented as red rectangles. In the middle lane are the email and issue management services because they are involved in both the new and current architectures.

The email service is realised by the Outlook software. Issue management is realised by the Jira system. No changes are foreseen in the way these services are realised in the future.

## 7.4 Implementation services and components

The implementation stage is the first place where considerable differences are visible in the way the application architecture is organised. This is depicted in Figure 7.5.

We start discussing this architecture by covering the common services first, situated in the middle lane of the diagram, and then we address particularities which are situated at the left and right sides of the diagram.

The issue management system is involved and is realised the same way as in the inception/evolution stage (Section 7.3). In the following sections the services and their realisation that have been already discussed will no longer be addressed.

File management & versioning is realised through SVN [2]. In the current appli-

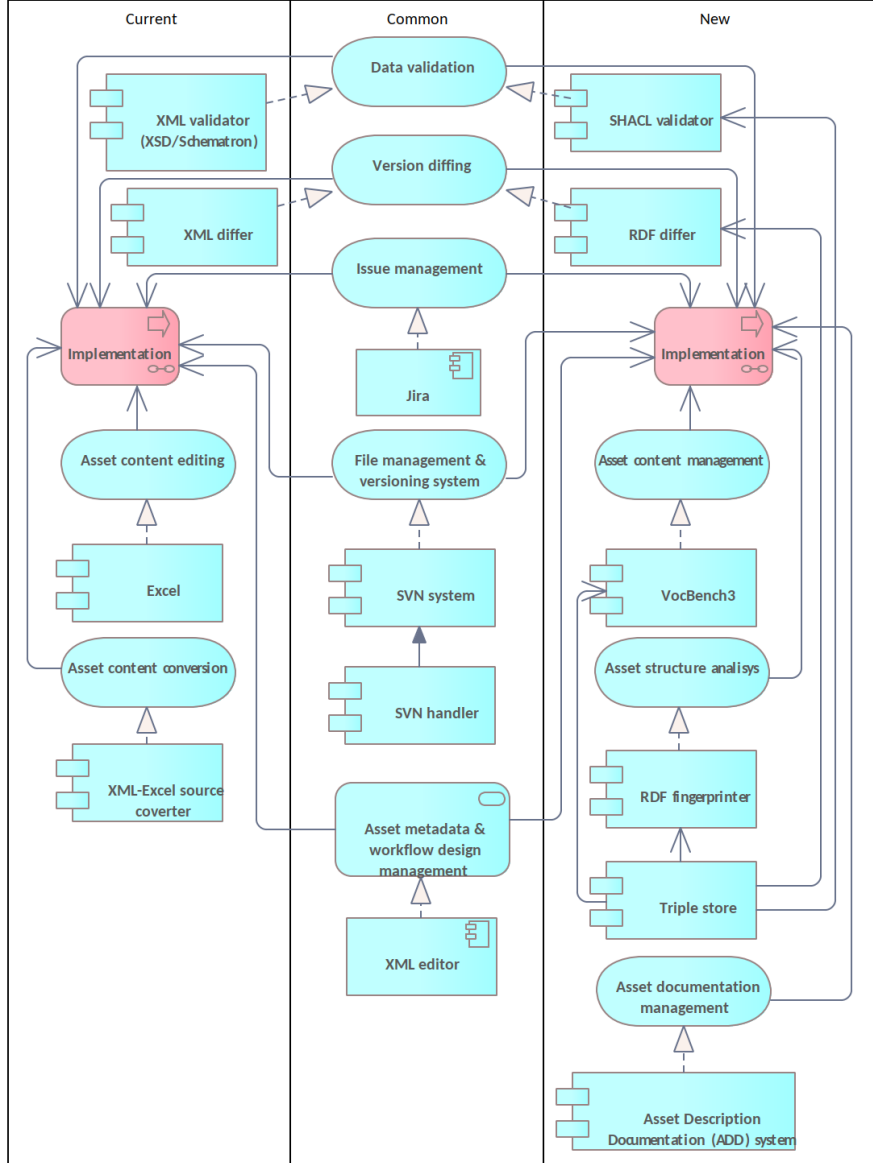


Figure 7.5: The application services and components that serve the current and new implementation stage

cation architecture this serves as a foundation for process automation, the triggers being any operations within the SVN repository. The mechanisms employed are similar to what an automation server is doing in the software development context:

triggering automatic building, testing and deploying, facilitating continuous integration and continuous delivery. No such automation server is, however, used due to infrastructure limitations that existed in the past and a custom SVN handler component was developed within the legacy workflow system.

In the new architecture, the possibility to continue the same practice of automating process executions based on SVN triggers is still there. Moreover, we recommend replacing the current SVN handler by an automation system, such as Jenkins [31] and Bamboo [4]. This suggestion is not depicted in the diagram because the current workflow SVN handler could still serve in the immediate future; however, for the long run, we recommend an alternative.

Asset metadata and workflow design management is centralised in a custom-built XML file called the "Vocabulary table". The vocabulary table is a file which contains descriptions of currently-administered digital assets, descriptions of generic process execution flows and decision rules for special assets. For this reason, we say the it is "workflow design management" and not the actual "workflow management", because we deal here with description and configuration of the execution paths and decision rules (plus asset metadata) which are enacted by the legacy workflow system. This aggregation of the two responsibilities complicates both the maintenance and evolution both shall be separated in the future.

Later, after adoption of the digital transformation proposed in this architecture, the asset metadata and workflow design management service need to be broken down into two parts: asset metadata management and workflow management. The metadata management can be realised through a DCAT [54] cataloguing software.

Workflow management could be taken over by any workflow management and execution engine that implements service orchestration<sup>2</sup> or choreography<sup>3</sup>.

This could be taken one step further towards a BPMN [53] execution engine such as Camunda BPM, Flowable and Bonita BPM. There is an installation and configuration overhead for such a system, but the investment is worth considering knowing that they ship with tools for creating workflow and decision models, operating de-

---

<sup>2</sup>In system administration, orchestration is the automated configuration, coordination, and management of computer systems and software. A number of tools exist for automation of server configuration and management, including Ansible, Puppet, Salt, Terraform, and AWS CloudFormation.

<sup>3</sup>Service choreography is a form of service composition in which the interaction protocol between several partner services is defined from a global perspective.

ployed models in production, and allowing users to execute workflow tasks assigned to them.

Now we should focus on the top of the middle lane, to the data validation and version diffing services. The components that realise them are split between the left and right lanes. This means that currently validation is performed using the XML validator and the version diffing is performed using XML differ, while in the new application, validation is realised using a SHACL [30] validator, and version diffing using an RDF differ. The SHACL validator and RDF differ are custom components developed for the new application context.

In the right lane, the asset content editing and the asset content conversion services were introduced in Section 7.2. They are the gateway for the asset authoring officers to implement change request cases. Content editing is realised through MS Excel running on the documentalists' workstations. Once the edit is finished, the updates are committed to the common SVN repository. The changes are picked up by the SVN handler in the legacy workflow system and trigger a conversion process from MS Excel to XML, and then back to MS Excel. The conversion is realised by another legacy component implemented to do exactly that. The resulted conversion is committed into the SVN repository again. This circular conversion, as explained in Section 6.7, is necessary because the main source of the assets is represented as CAT-XML, whereas MS Excel workbook is merely an extension that enables user-friendly editing capabilities.

In the new version of the implementation stage, the main source of the assets is represented as SRC-AP [12], a particular form of SKOS [36] and is serialised as RDF [55]. The editing is covered by the asset content management service, which is realised by the VocBench3 system [48]. When the authoring officer finishes the implementation of a request case, the content is exported from VocBench3 and committed into the SVN common repository. This commit triggers automatic SHACL validation, RDF diffing and structure analysis operations, which write the results back into the common repository for the editor to look at and verify the implementation correctness.

Asset structure analysis service is realised by the RDF fingerprinter component which is necessary for the quality assessment in this and the next stage of the asset life-cycle. Finally, asset documentation management service was described in Section 7.2.

The new RDF-based components rely mostly on the existence of an RDF triple store service, which is a database ensuring persistence, accessibility and data query. This triple store component is depicted not as realising any service in particular, but as serving other components: RDF fingerprinter, SHACL validator, RDF differ and VocBench3, so that they function as expected this way, creating a dependency between them. We use a generic label “triple store” here to express that it is of little importance what system is chosen in particular. Among the candidates we can mention are Fuseki, Virtuoso, GraphDB, StarDog, AnzoGraph,... the list can continue.

This brings us to the end of the implementation stage description which provided us with a parallel between the services and components as they are used currently and how the new application should be implemented.

## 7.5 Pre-release and validation services and components

The stages following the implementation in both the current and new life-cycle processes differ. In Section 6.11 we described that in the current life-cycle it is termed the pre-release stage while in the new one it is the validation stage. This difference is also reflected on the application architecture in Figure 7.6.

The current pre-release stage, on the left lane of the diagram, involves asset meta-data and workflow design management service, described in Section 7.4, due to the workflow automation involved in this stage.

The data validation was also described in Section 7.4. One thing that was not mentioned is that the validation relies on an XSLT [27] controller component which triggers the XML validator component, also the XSLT transformation component realising the data transformation service.

The data transformation service is used currently in the pre-release stage to format the RDF/XML content. Internally, this operation is known as SKOS “prettification”, i.e. making the SKOS-RDF/XML easier to work with. This operation will no longer be necessary in the new architecture and the XSLT transformation component will only be employed in the next stage: release.

The issue management, file management and versioning services are used in the

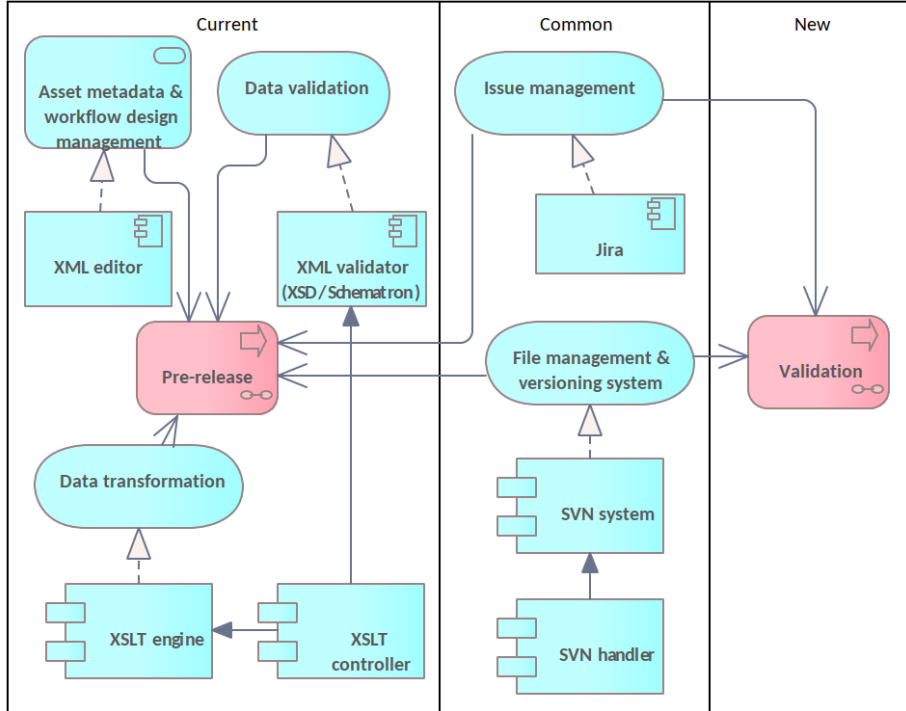


Figure 7.6: The application services and components that serve the current pre-release and the new validation stages

pre-release stage and are the only ones necessary in the validation stage. Therefore, they are placed in the middle, common, lane. These services have been detailed in Section 7.3 and 7.4. The validation stage requires nothing more than these because it is mostly a manual process executed by the validation officer who checks the validation artefacts generated previously in the implementation stage, as explained in Section 6.13.

## 7.6 Release services and components

The release stage is significantly unbalanced in terms of the services employed currently and what is foreseen in the new architecture: this is reflected in the application architecture depicted in Figure 7.7.

Currently, in the release stage described in Section 6.9, the asset(s) are mainly copied from one part of the common repository into a another which is dedicated to the

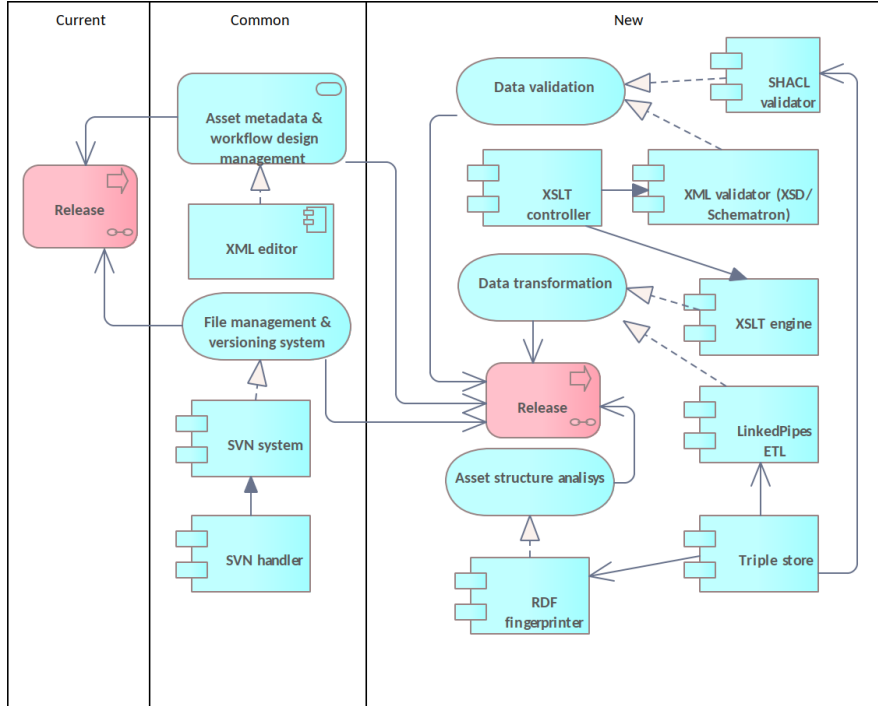


Figure 7.7: The application services and components that serve the current and new release stage

artefacts that follow to be published.

In the new release stage, in addition to the copying operation, all data transformation operations described in Section 6.10 are also executed leaving the (new) publication stage to deal with packaging and distribution mainly (from a technical perspective).

This is reflected in the new application architecture depicted in the right lane. The data transformation services are employed here, not only in their current form, dealing with various formats consumed by the current clients (XSD, MarkXML, GeoJSON, etc.), but also with complex transformation operations on RDF representation. The former is realised through the same XSLT engine component, while the latter is realised through a new component called LinkedPipes ETL [29, 28].

The situation is similar with the data validation service: both the current validation component based on XSD schemas and Schematron checks, and a new validation based on SHACL shape checks, are necessary. Both validation types are used because current data formats and new data formats needs to be formally verified before

being published. This validation, however, aims to ensure primarily that the transformation processes run correctly and, to a lesser effect, the content correctness which is already assessed in the previous validation stage. The asset structure analysis service was addressed in Section 6.12. Also, the new components that operate with RDF data require a triple store which is depicted in the left lane as serving the depending components.

## **7.7 Publication services and components**

The publication stage is the last one in the asset life-cycle process. Its application architecture is depicted in Figure 7.8. There is not much difference between the two, most services and components being placed in the middle lane, common to both.

New and specific to the publication stage is the impact assessment. The release notes generation service provides the basic communication artefacts about what has changed between two versions of an artefact and what is contained in a publication. To a large extent, it is based on the diffs generated in the implementation stage, but not limited to those. However, this has an impact on how the component functions because the diffs of two XML files are different from the diffs of two RDF files. The asset source format changes from XML to RDF in between the current and new architectures: therefore, the current implementation of the service cannot be reused and a new impact assessment generator shall be developed for the new application.

The following services have already been addressed in the previous sections and are not discussed here: management and versioning, e-mail, issue management and asset metadata and workflow design management.

The asset packaging service, placed in the middle lane of the diagram, is another service that is specific to the publication stage only. It is realised by the legacy package generator which assembles in various ways asset artefacts as necessary for partner dissemination systems and final consumers. This component shall be reused in the new application architecture in order to ensure continuity: however, it may be replaced in further digital transformation initiatives.

The last service adopted in the new application architecture is the package transmission service. The added value of this service is that the publication officer does not any more need to manually ingest the packages into the dissemination system. For a start, this service is realised by the new CERES component which is responsible



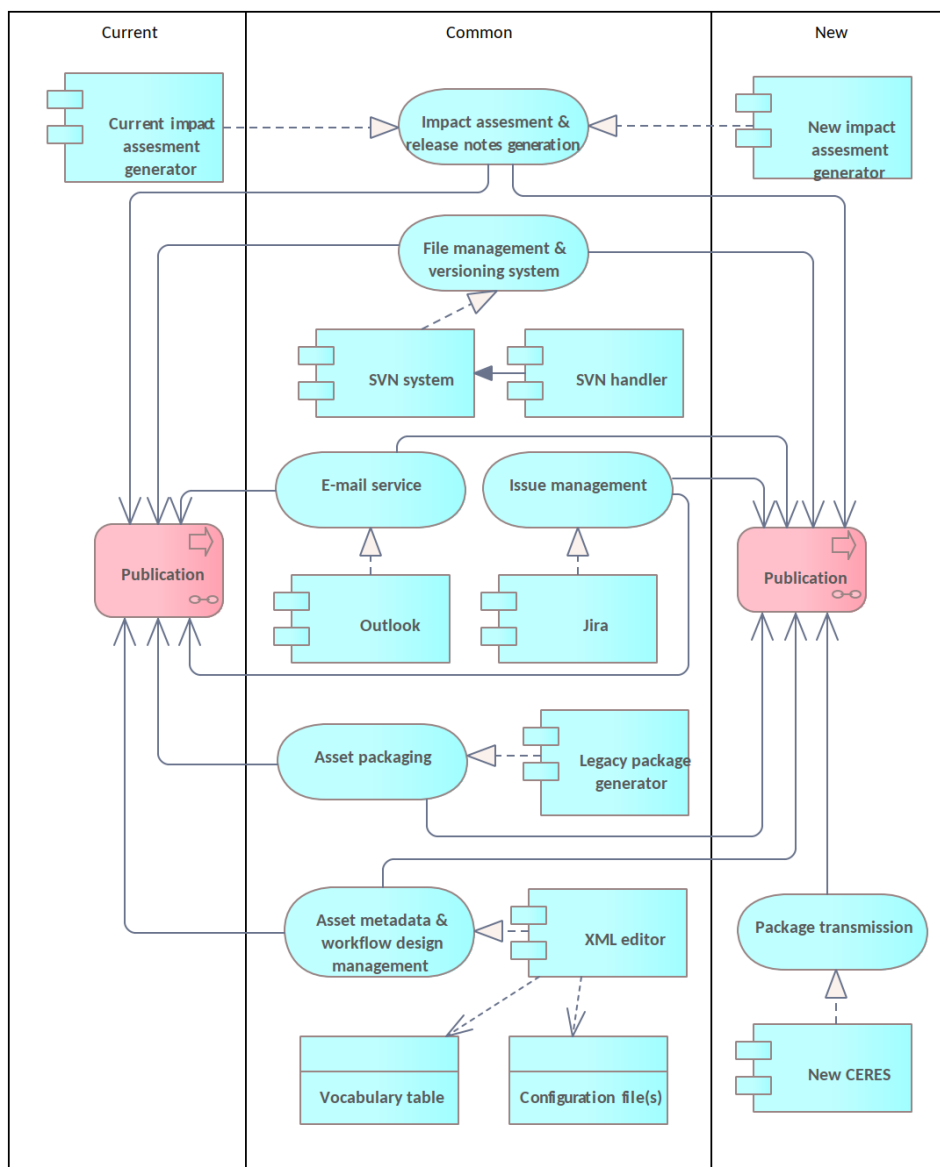


Figure 7.8: The application services and components that serve the current and new publication stage

for ingesting packages into Cellar. Later on, additional components can be created for ingesting packages into other dissemination systems such as Wikidata, Bartoc, ODP, etc.

This brings us to the end of the application architecture description. Next we will take a look at how the two applications are deployed in practice and the differences in the infrastructure.

# Chapter 8

## Technology architecture

This chapter covers the technology architecture (term from the ArchiMate language). It is used for modelling the underlying infrastructure and deployment of applications as software, servers, clusters, load balancers, etc. The next section presents a generic pattern which we substantiate in this architecture.

### 8.1 Prototypical technology structure

This section presents a generic technology architecture organisation, depicted in Figure 8.1, in order to ease interpretation of the diagrams that follow.

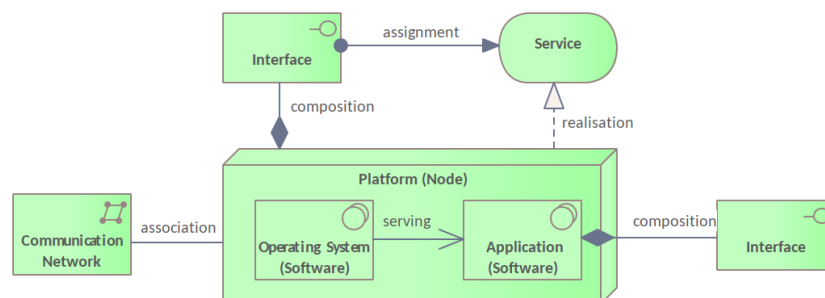


Figure 8.1: The prototypical technology view

The central concept in the technology view is the node. It represents a computational or physical resource that hosts, manipulates or interacts with other computational

or physical resources. Within a node, various software components are deployed, most foundational of which is the operation system. The node aggregates various software.

Depending on the chosen level of granularity, nodes may realise technology services which represent explicitly defined and exposed behaviour. Also, nodes and software may expose interfaces which represent points of access where a technology service offered by a node can be accessed. Interfaces constitute parts of the node: an interface may be assigned to a service.

Lastly, node are connected to the communication network which represents a set of structures that connects nodes for transmission, routing and reception of data. There may be one or more networks depending on the organisation security policies and on the internal infrastructure setup. These ArchiMate elements suffice to describe the current and new technology architectures.

## 8.2 Current technology architecture

The current SU infrastructure necessary to support the application life-cycle is depicted in Figure 8.2. It comprises four nodes for a private network and a controlled connection to the internet.

At the bottom of the diagram is the workstation node which represents the standard machine provided to the members of the SU team by the DIGIT infrastructure unit. The workstation runs Windows 10 [35] operating system and the following set of tools (at least) are installed on it: MS Excel, Outlook, Chrome (or another modern browser), an SVN client (Tortoise SVN is the default choice by the SU team), a specialised XML editor (XML Spy is the default), an FTP client of choice and an SSH client (Putty is the default). This set of software is used to perform all the necessary duties and enact the described business processes.

At the top of the diagram is depicted the standard service hosting offered by DIGIT. The provided services there are the e-mail service, a dedicated the SVN repository as a service and a set of Jira projects.

In addition, two identically-housed Linux servers are provided: one acting as the acceptance environment and the second as the production environment. They are called MDR Linux hosting nodes and they host the legacy workflow management system. This legacy system is based on multiple technologies which were added

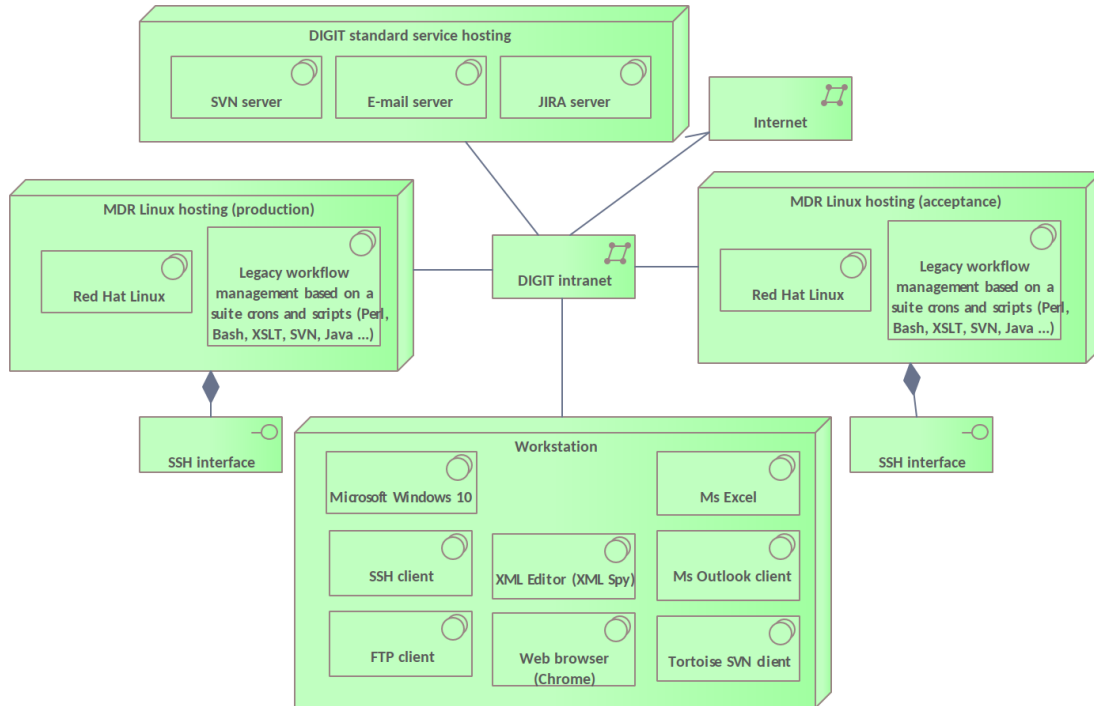


Figure 8.2: The technology structure that supports the current asset lifecycle

gradually and organically in the course of its development. It includes components written as Perl scripts, Bash scripts, many XSLT transformation style-sheets, commands relying on existence of an SVN client and Java libraries and scripts.

The MDR Linux hosting nodes serve as testing ground and as an operational environment. The nodes run RedHat Linux 7.7 operating system and each expose an SSH interface. These interfaces are used by the technical team. The technicians connect through it from their workstations and operate the legacy workflow management system remotely.

The communication network is secured as an intranet-authenticated connection provided by DIGIT. From the intranet, a controlled access to the Internet is available through an authentication proxy.

This completes the description of the current infrastructure architecture and sets the baseline for the new architecture described in the next section.

### 8.3 New technology architecture

The new infrastructure architecture is depicted in Figure 8.3. What are the same as in the current one are the structure of the workstation node, the intranet and the Internet connections, as well as standard hosted services to which are added VocBench3 and the Adobe FrameMaker server which are necessary for realising the asset description documentation (ADD) service, in addition to a few other important services within SU, among which is the generation of the inter-institutional style-guide.

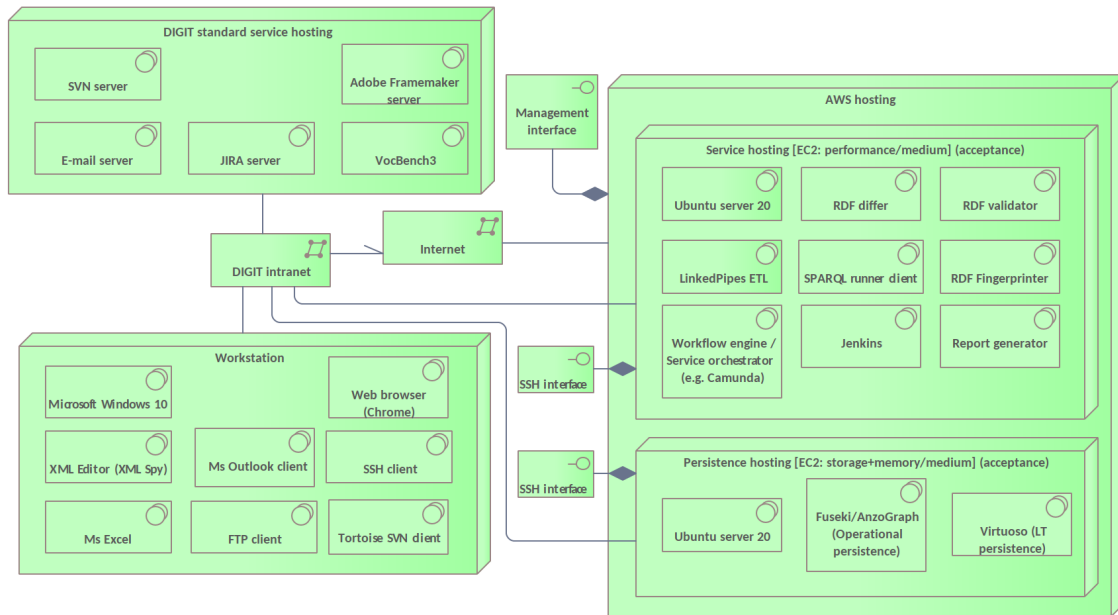


Figure 8.3: The technology structure that supports the new asset lifecycle

What differs significantly between the two is the disappearance of the hosted nodes (left and right side of Figure 8.2) and their replacement with dedicated Amazon Web Service (AWS) hosting, depicted as a large node on the right side of Figure 8.3. This node exposes a management interface which provides the possibility of configuring and launching multiple virtual servers (called EC2 instances) of various sizes and performance capabilities. This puts an administration overhead onto the SU team, but also frees it from a large number of constraints and experiences in the past years, this way smoothing the path for this and future digital transformations.

In the new architecture, to support the application architecture described in Section

7, two instances are necessary: one optimised for performance and another optimised for persistence. The first one hosts the operational application components, while the second one triple stores potentially other types of databases. This initial separation into two could be further optimised by moving some services to dedicated EC2 instances if deemed necessary. Also, these two instances are considered an acceptance environment where the new application is assembled and tested. Once it is ready to move into production, an identical copy of EC2 instances is created and those are further used for production operations.

The EC2 instance at the top, marked for performance, runs Ubuntu operating system version 20 (server edition) and has installed the following software : RDF differ (custom-built component), RDF validator (custom-built component), RDF fingerprinter (custom-built component) and Report generator (custom-built component), LinkedPipes ETL, SPARQL runner client (custom-built component), Jenkins automation system and Camunda BPM.

The EC2 instance at the bottom, marked for persistence, runs Ubuntu as well, and hosts initially two triple stores: Fuseki or AnzoGraph, which are optimised for operational performance, and an instance of Virtuoso triple store which is meant for long-term preservation of RDF data.

Each of the EC2 instances expose an SSH interface which can be accessed by the technical team to configure, manipulate and operate the servers, just like in the currently-hosted MDR Linux server.

## Chapter 9

# Technology deployment proposal

This chapter describes the detailed infrastructure organisation necessary for supporting the new application architecture of the asset publishing life-cycle. This represents the initial setup for starting the transition towards the new architecture.

We split our presentation into manageable chunks but it is important to bear in mind that they are not separated parts but views of a larger whole. The views do not correspond to the asset publishing life-cycle but are given some generic names based on the presented components. These views are mainly focused on the organisation within the AWS nodes depicted in Figure 8.3 but can occasionally touch on other parts of the technology structure.

In the current proposal we adopt a service oriented approach as motivated in 2.3. We assume the reader has a basic understanding in how Docker platform [43] functions because each software element in the diagrams below represents a service deployed and executed as a Docker container. The artefact elements marked with “data volume” label represent attached Docker volumes.

### 9.1 VocBench3 export and automation services

The aim of this section is to explain how to setup the infrastructure so, that the export from VocBench3 can be triggered by a software component through an API call or a SU team member through the user interface. Even more so, this export shall automatically trigger other components, which are responsible of diffing, validation



and fingerprinting the exported content.

VocBench3 (the blue block in Figure 9.1) is an application component hosted under the standard DIGIT service agreement. This means that this software is accessible through its interfaces and no direct configuration is possible. It is realised by the VocBench3 Angular client which exposes a user interface and by the Semantic Turkey middleware (ST) server which serves the Angular application. The Semantic Turkey middleware exposes an API interface that can be remotely called from external by services outside the DIGIT hosting, i.e. AWS instances under the management of SU.

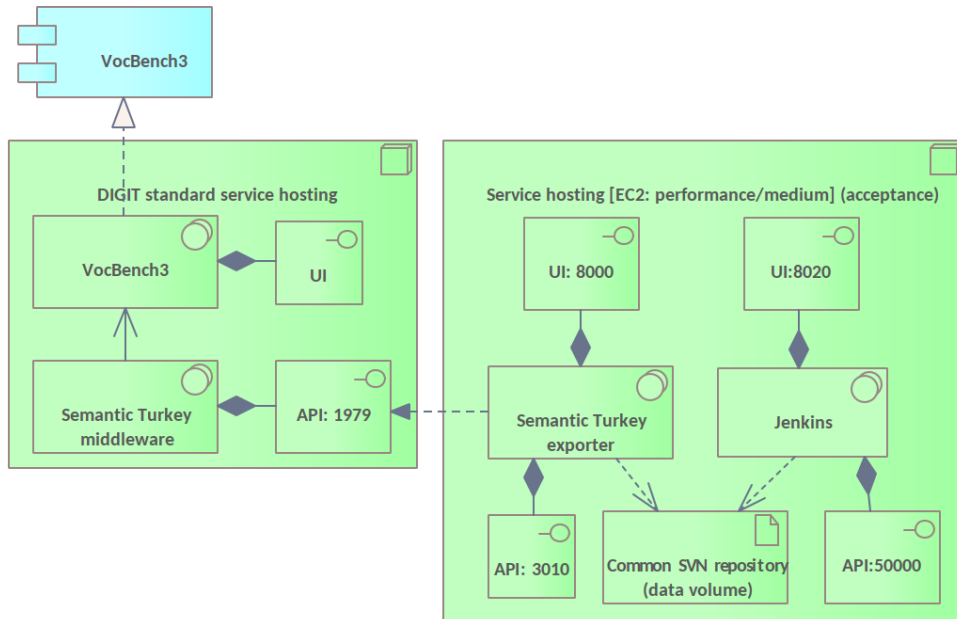


Figure 9.1: The technology components for exporting asset sources from VocBench3 into the common SVN repository

Semantic Turkey exporter component shall be configured according to the Semantic Turkey installation offered by DIGIT (port, address and access rights). This component exposes a user interface and an API which allow exporting a selected project from VocBench3 and committing it into the common SVN component. This operation could also be manually performed before such exporter is in place.

An instance of Jenkins automation server must be deployed and configured to listen

to changes in the common SVN repository on preselected projects. It can be configured through the user interface or configuration files at the runtime. It also must be configured to trigger execution of RDF differ, RDF validator and RDF fingerprinter on the exported asset(s). Once an export is committed into the SVN repository Jenkins will automatically trigger chain execution of the preconfigured services.

The deployment organisation of the validation, diffing and fingerprinting services is described in the next section.

## 9.2 Validation services

In this section we explain how SHACL validation, RDF fingerprinting and RDF differ components are deployed depicted in Figure 9.2. Each of the tree components is realised by a software named with the same name. Each of these software exposes two interfaces: one is the user interface and another is an API. The API represents the potential for being operated by an automation server or an service orchestrator. We omit to picture the AWS node for service hosting as all the mentioned elements belong to it.

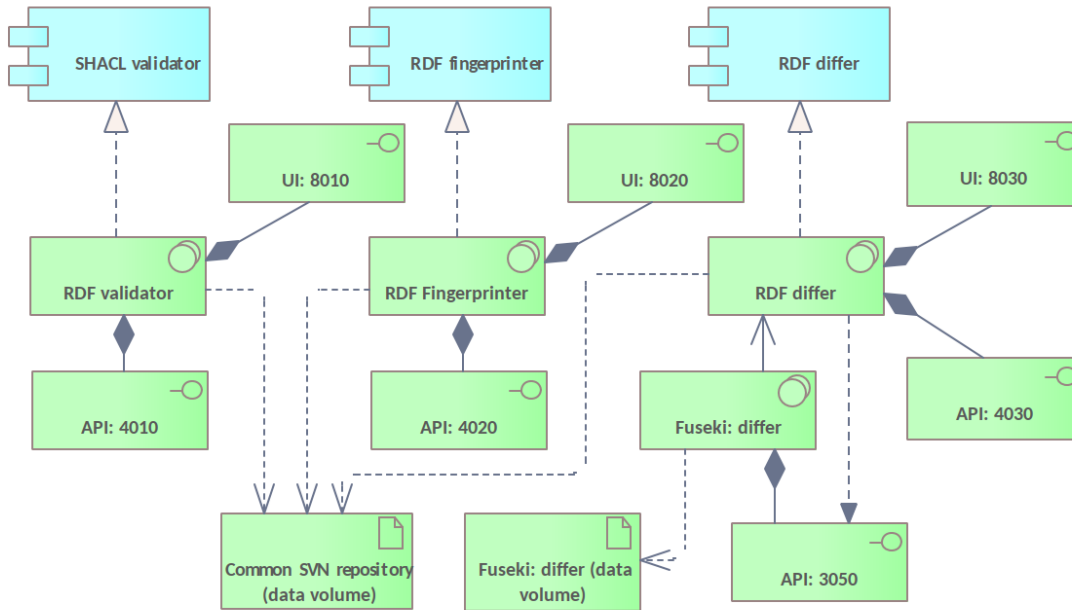


Figure 9.2: The technology components for generation of assessment artefacts

The RDF validator, RDF fingerprinter and RDF differ are deployed as Docker containers and all mount and have access to the common SVN repository volume. Doing so grants direct access to the stored files and provides the possibility to write and commit the results of their execution. For example the SHACL validator can access a file in the repository and write the validation report into the repository, next to the validated file. The RDF differ, in a similar manner, can load two versions of the same file from the common repository and compute the diff between them writing it back next to the latest version of it. The same holds for the RDF fingerprinter.

The RDF differ, does not run alone, it requires an instance of a running triple store, and in this case we select Fuseki as a tested solution. Another triple store can be chosen at a later stage. The reason for this is that the RDF diffing is executed through a series of SPARQL queries which can be executed only in a triple store. So in Figure 9.2 “Fuseki:differ” software is a Fuseki instance serving the RDF differ exclusively and is accessible through its API. The Fuseki instance persists its data outside the container in a dedicated volume mounted to it.

The port configuration suggested here is selected to be conflict-free and does not constitute a hard requirement and can be changed as the situation requires it.

### 9.3 Transformation services

This section covers the software components necessary in the release stage when the RDF asset sources are transformed into various release artefacts. The infrastructure components and their interrelations are presented in Figure 9.3.

The key element is the LinkedPipes ETL application component (in light blue on the top of the diagram), which realises the RDF-based transformation and conversion of data. It is realised by a software element with the same name (in green below). This software exposes four technology interfaces, one of which is a user interface and the other three are technical. The FTP interface is very useful for debugging transformation processes while the other two for triggering and controlling the transformation process executor and the storage controller.

LinekdPipes ETL shall use the externally mounted SVN repository that serves as source for the transformation processes inputs and a place to write the final outcomes. For execution of the intermediary transformation operations and processing it needs another volume which is mounted to it. In addition it uses also a triple store

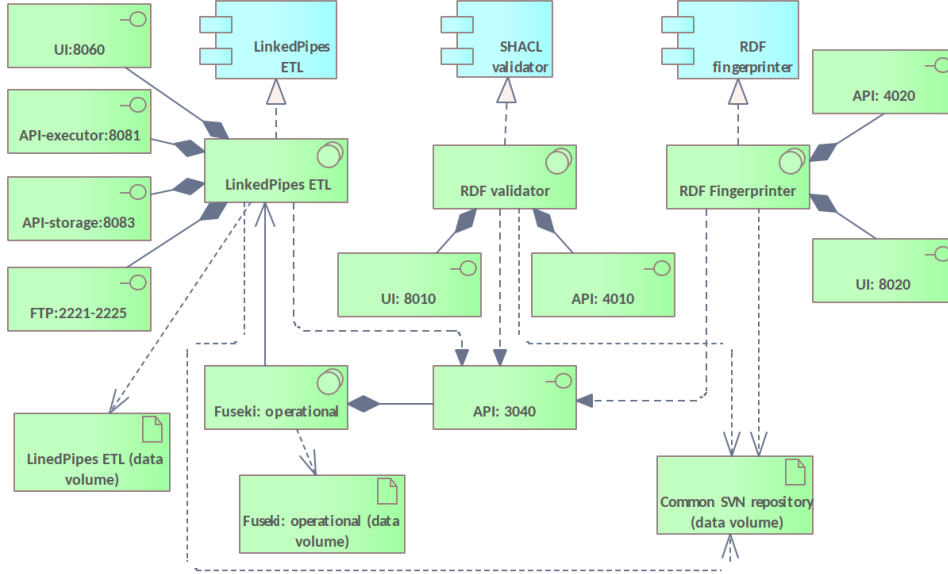


Figure 9.3: The technology components organisation for generation of release artefacts

for the same, operational, reasons. We propose to use a Fuseki instance dedicated to LinkedPipes ETL operations, which we call “Fuseki: operational”. This Fuseki instance also shall use an externally mounted data volume for convenience and ease of management.

As explained in the Section 6.14, the transformation processes need to be controlled. Therefore the validation services described above are repeated here again to show the connection with the transformation service. The validation services have access to the “Fuseki: operational” instance and to the common SVN repository. Like this, a validation procedure can be triggered at any stage of the transformation process. The RDF differ is omitted from this figure because it is relevant in verifying the content implementation completeness and correctness and required human involvement. The SHACL validation and fingerprinting services can be used in a fully automated and semi-automated scenarios, and requiring little to no human intervention.

## 9.4 Reporting services

This section covers the software components necessary for producing human readable reports on various aspects in asset publishing life-cycle. The infrastructure components and their interrelations are presented in Figure 9.4.

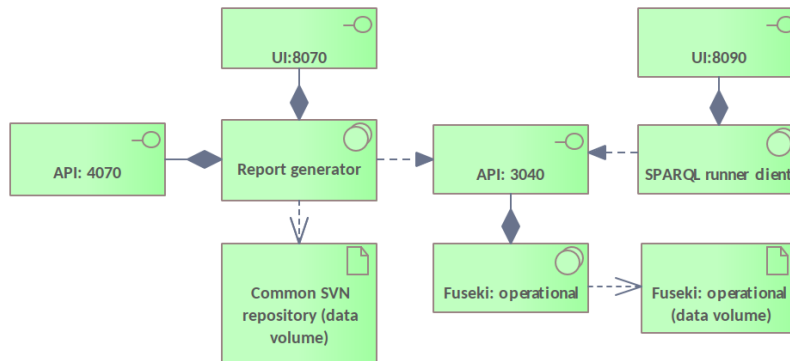


Figure 9.4: The technology components for generation of human readable artefacts

The report generator is a software component which generates various reports in human readable formats such as HTML, PDF, Excel, DOCX. It does so by the virtue of predefined templates and by accessing data from the common SVN repository and from the Fuseki operational triple store. It exposes an user interface but can also access via an API.

A very useful tool, especially fro the technical staff is a SPARQL client which can be used to execute SPARQL queries directly on SPARQL endpoints and display the results. We call it “SPARQL runner client”. This software comes in handy when predefined queries need to be executed remotely in an automated or manual fashion outside the triple store facilities.

## 9.5 Service orchestration

This section covers the software components necessary for service orchestration across all steps of the asset publishing life-cycle. Adoption of a service orchestration application component was discussed in Section 7.4. The infrastructure components and their interrelations are presented in Figure 9.5.

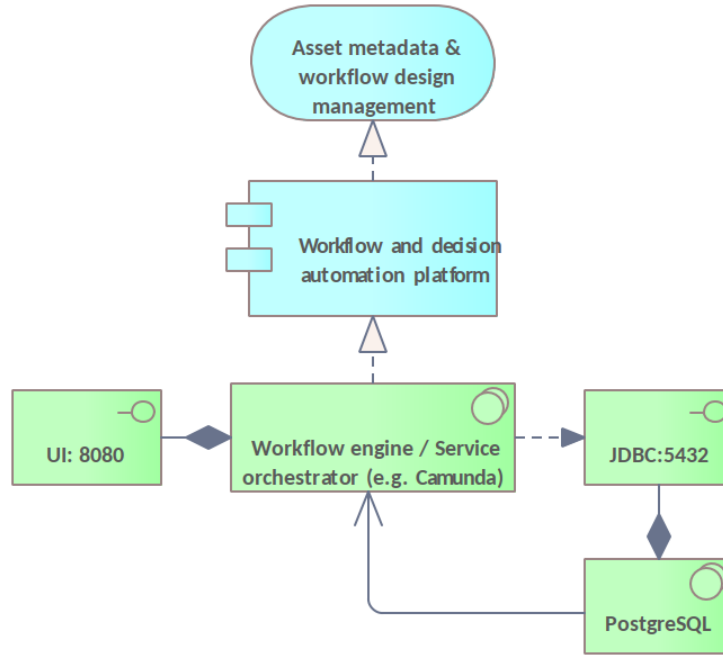


Figure 9.5: The technology components for generation of human readable artefacts

In system administration, orchestration is the automated configuration, coordination, and management of computer systems and software. This can be done by designing and then executing a BPMN process. Business Process Model and Notation (BPMN) is a graphical representation for specifying business processes in a business process model. The objective of BPMN is to support business process management, for both technical users and business users, by providing a notation that is intuitive to business users, yet able to represent complex process semantics.

We propose deployment of an instance of Camunda BPM server, which ships with tools for creating workflow and decision models, operating deployed models in production, and allowing users to execute workflow tasks assigned to them. It provides a BPMN compliant workflow engine and a Decision Model and Notation (DMN) compliant decision engine.

To support its functionality, Camunda needs a PostgreSQL database, which can be deployed in a sibling Docker container. Once Camunda BPM is deployed, BPMN workflows can be designed to access and trigger any services available on the AWS instances or any standard services offered by DIGIT. This will result in business staff

being able to follow and execute entirely these workflows because the technicalities will be entirely hidden.

# Chapter 10

## Conclusions

This document presented the architectural stance for the asset publication life-cycle as is currently employed by the SU and as we recommend it to become.

Through the development of this architecture we bring clarity to the SU management of who are the main stakeholders, what their interests and drivers are and what issues and solutions are associated to those motivations. Also we explicitly describe the internal processes, events and roles answering questions concerning who shall do what and when. Valuable especially for the technical staff is the application architecture which answers the questions about what application service and capability supports which process in the asset publishing life-cycle, and infrastructure related questions of what components are deployed in which nodes and how they communicate between each other.

This document constitutes a way to move forward with the digital transformation of the SU given its current interinstitutional context, management goals and demands from third parties.

We aim to guide transitions in the asset source representation from the current, XML-based source, towards a new, RDF-based source.

The data quality is addressed in the current architecture through introduction of manual and automatic verification and validation steps operating at both the level of form and the meaning.

This architecture sets the trend for modernisation of the currently employed appli-



cation. It is not, however, addressing the entire digital transformation in order to prevent disruption in the current production system. Rather a part of the application is foreseen to evolve, that responsible for editing the asset content, and the rest can be addressed in a subsequent step as a natural followup.

This architecture reorganises the business processes aiming to optimise process workflow reducing the bottlenecks and increasing the speed of the overall life-cycle process. The aim is to achieve the performance of “overnight publications”.

## 10.1 Summary

We presented in Section 2 the context of the current work given by the EU decisions and directives towards the semantic web technologies, open data and digital re-use of public sector information, along with implementation of a single digital gateway. The description of the state of play sets the baseline technical assessment which is extended by a recommendation of a joint trend towards the semantic web technologies and service oriented architecture.

The architecture proposed here consists of four layers: motivation in Section 4, business in Section 5 and 6, application in Section 7 and technology in 8. And we finally conclude with this section.

## 10.2 Limitations and future work

It would be highly valuable to adopt internally in the SU but also in a broader context a data governance policy. For that additional roles would have to be adopted in the organisation, a set of policies and reports enforced and some measurement and indicator measurement capabilities implemented.

The new components that should be implemented for the new application life-cycle need to be described in terms of functional and non functional requirements, design choices and technology to be adopted. This is an essential input for the software development team which will deal with the implementation of those software components.

The new components that should be developed in order to support the new application are: RDF validation service, RDF fingerprinting service, RDF diffing service,

RDF based impact assessment service, report generation service and asset cataloguing and metadata management service. Among the new services shall also be included a document management service, content management service and a workflow management and execution service.

A clear and detailed account of the data structures and data objects that are operated in the application shall be provided. This is important in order to ensure data consistency, validation policies and control the flows between processes.

No technical capabilities are currently foreseen for process execution monitoring, log analysis and performance measurement. This constitutes a valuable technical capability for both clearly diagnosing and respond in an agile manner to a situation, and long term observations and derivation of optimisation insights.

## 10.3 Final word

In this document we propose the first step towards a modern enterprise-level application that streamlines the process of asset publication life-cycle forth both the internal staff of the SU and for external partners involved in the process.

We envisage a service oriented and semantically enriched system that operates in a cloud infrastructure and providing seamless experience to all involved parties in performing their duties and responsibilities in a coordinated asset publishing process. Such as system can constitute a cornerstone for management, publication and dissemination of public sector reference data bringing the single digital gateway one step closer to reality.

# Bibliography

- [1] Adobe. FrameMaker, 1995. URL <https://www.adobe.com/products/framemaker.html>.
- [2] Apache Software Foundation. Apache Subversion, 2000. URL <https://www.office.com/>.
- [3] P. Archer, L. Bargiotti, M. D. Keyzer, S. Goedertier, N. Loutas, and F. V. Geel. Report on high-value datasets from eu institutions. Deliverable SC17DI06692, European Commission, 2014.
- [4] Atlassian. Bamboo, 2002. URL <https://www.jenkins.io/>.
- [5] Atlassian. Jira Software, 2002. URL <https://www.atlassian.com/software/jira>.
- [6] D. Beckett. RDF/xml syntax specification (revised). W3C recommendation, W3C, Feb. 2004. <http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/>.
- [7] M. Brandner, M. Craes, F. Oellermann, and O. Zimmermann. Web services-oriented architecture in production in the finance industry. *Informatik-Spektrum*, 27(2):136–145, 2004.
- [8] T. Bray, M. Sperberg-McQueen, E. Maler, F. Yergeau, and J. Paoli. Extensible markup language (XML) 1.0 (fifth edition). W3C recommendation, W3C, Nov. 2008. <http://www.w3.org/TR/2008/REC-xml-20081126/>.
- [9] D. Brickley and R. Guha. RDF schema 1.1. W3C recommendation, W3C, Feb. 2014. <http://www.w3.org/TR/2014/REC-rdf-schema-20140225/>.

- [10] G. Carothers and E. Prud'hommeaux. RDF 1.1 turtle. W3C recommendation, W3C, Feb. 2014. <http://www.w3.org/TR/2014/REC-turtle-20140225/>.
- [11] K. Channabasavaiah, K. Holley, and E. Tuggle. Migrating to a service-oriented architecture. *IBM DeveloperWorks*, 16:727–728, 2003.
- [12] E. Costetchi. SRC-AP-VB3: an application profile for metadata assets maintained in VocBench3. Recommendation, Publications Office of the European Union, September 2019.
- [13] E. Costetchi and J. Delahousse. METADATA ENCODING AND TRANSMISSION STANDARD. Primer and reference manual, Digital Library Federation, September 2010.
- [14] E. Costetchi and J. Delahousse. SKOS-AP-EU: an application profile for publishing SKOS-based metadata assets. Recommendation, Publications Office of the European Union, September 2019.
- [15] D. Day, M. Priestley, and D. Schell. Introduction to the darwin information typing architecture. *IBM corporation*, 2005.
- [16] D. Day, K. J. Eberlein, R. D. Anderson, and G. Joseph. Darwin Information Typing Architecture (DITA) Version 1.2. OASIS Standard, OASIS, dec 2010. <http://docs.oasis-open.org/dita/v1.2/spec/DITA1.2-spec.html>.
- [17] European Parliament and the Council. Decision No 456/2005/EC of the European Parliament and of the Council of 9 March 2005 establishing a multi-annual Community programme to make digital content in Europe more accessible, usable and exploitable (Text with EEA relevance). *OJ*, L 79:1–8, 2005. CELEX:32005D0456.
- [18] European Parliament and the Council. Directive 2013/37/EU of the European Parliament and of the Council of 26 June 2013 amending Directive 2003/98/EC on the re-use of public sector information Text with EEA relevance. *OJ*, L 175: 1–8, 2013. CELEX:32013L0037.
- [19] European Parliament and the Council. Decision (EU) 2015/2240 of the European Parliament and of the Council of 25 November 2015 establishing a programme on interoperability solutions and common frameworks for European public administrations, businesses and citizens (ISA2 programme) as a means

- for modernising the public sector (Text with EEA relevance). *OJ*, L 318:1–16, 2015. CELEX:32015D2240.
- [20] European Parliament and the Council. Regulation (EU) 2018/1724 of the European Parliament and of the Council of 2 October 2018 establishing a single digital gateway to provide access to information, to procedures and to assistance and problem-solving services and amending Regulation (EU) No 1024/2012 (Text with EEA relevance.) . *OJ*, L 295:1–38, 2018. CELEX:32018R1724.
- [21] European Parliament and the Council. Directive (EU) 2019/1024 of the European Parliament and of the Council of 20 June 2019 on open data and the re-use of public sector information. *OJ*, L 172:56–83, 2019. CELEX:32019L1024.
- [22] E. Francesconi, M. W. Küster, P. Gratz, and S. Thelen. The ontology-based approach of the publications office of the eu for document accessibility and open data services. In *International Conference on Electronic Government and the Information Systems Perspective*, pages 29–39. Springer, 2015.
- [23] O. Group et al. Soa reference architecture. *Technical Standard*, available at [http://www.opengroup.org/soa/source-book/soa\\_refarch/](http://www.opengroup.org/soa/source-book/soa_refarch/), accessed in, pages 12–12, 2016.
- [24] R. Guha and D. Brickley. RDF vocabulary description language 1.0: RDF schema. W3C recommendation, W3C, Feb. 2004. <http://www.w3.org/TR/2004/REC-rdf-schema-20040210/>.
- [25] P. Hayes. RDF semantics. W3C recommendation, W3C, Feb. 2004. <http://www.w3.org/TR/2004/REC-rdf-mt-20040210/>.
- [26] G. Hillenius. Free and open source software across the eu. *IFOSS L. Rev.*, 5: 153, 2013.
- [27] M. Kay. XSL transformations (XSLT) version 3.0. W3C recommendation, W3C, June 2017. <https://www.w3.org/TR/2017/REC-xslt-30-20170608/>.
- [28] J. Klímek and P. Škoda. Linkedpipes etl in use: practical publication and consumption of linked data. In *Proceedings of the 19th International Conference on Information Integration and Web-based Applications & Services*, pages 441–445, 2017.

- [29] J. Klímek, P. Škoda, and M. Nečaský. Linkedpipes etl: Evolved linked data preparation. In *European Semantic Web Conference*, pages 95–100. Springer, 2016.
- [30] H. Knublauch and D. Kontokostas. Shapes constraint language (SHACL). W3C recommendation, W3C, July 2017. <https://www.w3.org/TR/2017/REC-shacl-20170720/>.
- [31] Kohsuke Kawaguchi. Jenkins, 2011. URL <https://www.jenkins.io/>.
- [32] A. Ledl and J. Voß. Describing knowledge organization systems in bartoc and jskos. *e-prints in library and informations science*, 2016.
- [33] Microsoft. Outlook, 2002. URL [www.microsoft.com/outlook](http://www.microsoft.com/outlook).
- [34] Microsoft. Excel 2013, 2013. URL <https://www.office.com/>.
- [35] Microsoft. Windows 10, 2015. URL <http://www.windows.com>.
- [36] A. Miles and S. Bechhofer. SKOS simple knowledge organization system reference. W3C recommendation, W3C, Aug. 2009. <http://www.w3.org/TR/2009/REC-skos-reference-20090818/>.
- [37] J. Paoli, F. Yergeau, M. Sperberg-McQueen, T. Bray, E. Maler, and J. Cowan. Extensible markup language (XML) 1.1 (second edition). W3C recommendation, W3C, Aug. 2006. <http://www.w3.org/TR/2006/REC-xml11-20060816/>.
- [38] B. Parsia, P. Patel-Schneider, and B. Motik. OWL 2 web ontology language structural specification and functional-style syntax (second edition). W3C recommendation, W3C, Dec. 2012. <http://www.w3.org/TR/2012/REC-owl2-syntax-20121211/>.
- [39] P. Patel-Schneider and P. Hayes. RDF 1.1 semantics. W3C recommendation, W3C, Feb. 2014. <http://www.w3.org/TR/2014/REC-rdf11-mt-20140225/>.
- [40] P. Patel-Schneider, B. Parsia, and B. Motik. OWL 2 web ontology language structural specification and functional-style syntax. W3C recommendation, W3C, Oct. 2009. <http://www.w3.org/TR/2009/REC-owl2-syntax-20091027/>.

- [41] D. Peterson, A. Malhotra, S. Gao, M. Sperberg-McQueen, P. V. Biron, and H. Thompson. W3C xml schema definition language (XSD) 1.1 part 2: Datatypes. W3C recommendation, W3C, Apr. 2012. <http://www.w3.org/TR/2012/REC-xmlschema11-2-20120405/>.
- [42] G. Schreiber and F. Gandon. RDF 1.1 XML syntax. W3C recommendation, W3C, Feb. 2014. <http://www.w3.org/TR/2014/REC-rdf-syntax-grammar-20140225/>.
- [43] Solomon Hykes. Docker, 2013. URL <http://www.docker.com>.
- [44] Sparx Systems. Enterprise Architect, 2000. URL <https://sparxsystems.com/products/ea/>.
- [45] M. Sporny, D. Longley, G. Kellogg, M. Lanthaler, and N. Lindström. Json-ld 1.0. *W3C recommendation*, 16:41, 2014.
- [46] M. Sporny, D. Longley, G. Kellogg, M. Lanthaler, P.-A. Champin, and N. Lindström. JSON-LD 1.1—a JSON-based serialization for Linked Data. W3C recommendation, W3C, 2020. <https://www.w3.org/TR/json-ld11/>.
- [47] A. Stellato, A. Turbati, M. Fiorelli, T. Lorenzetti, E. Costetchi, C. Laaboudi, W. Van Gemert, and J. Keizer. Towards vocbench 3: pushing collaborative development of thesauri and ontologies further beyond. In *17th European Networked Knowledge Organization Systems Workshop, NKOS 2017*, volume 1937, pages 39–52. CEUR-WS, 2017.
- [48] A. Stellato, M. Fiorelli, A. Turbati, T. Lorenzetti, W. van Gemert, D. Dechand, C. Laaboudi-Spoiden, A. Gerencsér, A. Waniart, E. Costetchi, et al. Vocbench 3: A collaborative semantic web editor for ontologies, thesauri and lexicons. *Semantic Web*, (Preprint):1–27, 2020.
- [49] The Open Group. *The TOGAF Standard, Version 9.2*. The Open Group, 2018.
- [50] The Open Group. *ArchiMate 3.1 Specification*. The Open Group, 2019.
- [51] A. T. Velte, T. J. Velte, and R. Elsenpeter. Cloud computing: A practical approach, 2019.
- [52] D. Vrandečić and M. Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.

- [53] S. A. White. Introduction to bpmn. *Ibm Cooperation*, 2(0):0, 2004.
- [54] P. Winstanley, A. Perego, S. Cox, D. Browning, R. Albertoni, and A. G. Beltran. Data catalog vocabulary (DCAT) - version 2. W3C recommendation, W3C, Feb. 2020. <https://www.w3.org/TR/2020/REC-vocab-dcat-2-20200204/>.
- [55] D. Wood, R. Cyganiak, and M. Lanthaler. RDF 1.1 concepts and abstract syntax. W3C recommendation, W3C, Feb. 2014. <http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/>.