V4Design

Visual and textual content re-purposing FOR(4) architecture, Design and virtual reality games

H2020-779962

D5.1

Ontology network and reasoning framework

Dissemination level:	Public
Contractual date of delivery:	Month 10, 31/10/2018
Actual date of delivery:	Month 10, 31/10/2018
Workpackage:	WP5: Content integration, retrieval and presentation
Task:	T5.1: Semantic content representation T5.2: Semantic integration and reasoning
Type:	Report
Approval Status:	Final version
Version:	3.0
Number of pages:	62
Filename:	D5.1_V4Design_OntologyNetworkAndReasoningFramework_20181031_v3.0.pdf

Abstract

This deliverable documents the semantic models for mapping the V4Design-pertinent conceptualisations on ontology-related constructs. In addition, it describes the functionality of the first version of semantic integration and reasoning techniques. First, the purpose, scope, intended users and the requirements of the ontologies as identified at this phase of the project are described. Their specification has been driven by the WP7 initial user requirements identified for the individual scenarios, as well as by the dependencies incurring from the interaction with the WP3 and WP4 analysis components and the WP6 functionality aspects. Second, the literature is reviewed, covering both state of the art languages for

formal knowledge representation and existing ontologies covering domains and requirements relevant to those of V4Design. Third, the current status of the V4Design ontologies is described, discussing the main entities they comprise. Fourth, the basic principles that underpin the first preliminary version of the WP5 reasoning framework towards reasoning and interpretation are described. Last, the report presents examples of the created annotation models.

co-funded by the European Union

History

Version	Date	Reason	Revised by
0.0	12/09/2018	ToC creation	Georgios Meditskos, Stefanos Vrochidis
0.2	5/10/2018	User requirement analysis and competency questions creation	Eleni Kamateri
0.3	8/10/2018	State of the art analysis	Eleni Kamateri, Georgios Meditskos
0.5	10/10/2018	Initial annotation models and reasoning	Georgios Meditskos
0.6	14/10/2018	An example of the annotation pipeline using the simulation example	Georgios Meditskos
1.0	15/10/2018	First draft circulated to WP5-related partners for comments	Georgios Meditskos
2.1	17/10/2018	Pre-final draft sent for internal review	Georgios Meditskos
2.2	23/10/2018	Internal review comments	Simon Mille (UPF)
3.0	25/10/2018	Preparation of the final draft	Georgios Meditskos, Stefanos Vrochidis

Author list

Organization	Name	Contact Information
CERTH	Georgios Meditskos	gmeditsk@iti.gr
CERTH	Eleni Kamateri	ekamater@iti.gr
CERTH	Stefanos Vrochidis	stefanos@iti.gr

Executive Summary

The present deliverable reports mainly on the work carried out within T5.1 and T5.2, relevant to the development of the V4Design ontologies and the representation and mapping of content on ontological entities. In addition, it describes the first preliminary framework towards reasoning.

More specifically, the present deliverable presents the current content of the V4Design ontologies and the methodology adopted to build them. Based on the requirements set forth by WP7 and the dependencies incurring from the interaction with the other WPs (WP6), the purpose, scope, intended users and uses, and the requirements of the V4Design ontologies were identified. These specifications, along with the modelling insights from the relevant literature, served as guidelines for building the first version of the V4Design ontologies that currently comprises modules for capturing the analysis results (metadata) of other V4Design modules, such as aesthetics, localisation of buildings and objects, 3D model attributes, named concepts and entities derived from text analysis. All this information is used to build the V4Design knowledge graphs that capture and interlink metadata for the derived assets.

In addition, we present a preliminary version of the reasoning layer whose purpose is to enrich the supported semantics and metadata both at the terminological level, by defining additional class and property axioms, and at the assertional level by incorporating inference rules. The additional inference capabilities will afford the derivation of interpretations that are useful at query-time, i.e. when the end users will perform queries to retrieve relevant assets from the V4Design platform.

The work presented within this document presents the preliminary version of the V4Design ontologies, reasoning and interpretation framework. More elaborate ontology-based interpretation and reasoning tasks will be tackled in future versions of the framework and reported in upcoming deliverables.

Abbreviations and Acronyms

BIM	Building Information Modelling
CQ	Competency Question
DCMI	Dublin Core Metadata Initiative
DL	Description Logics
DnS	Descriptions and Situations
EDM	Europeana Data Model
GIS	Geographic Information Systems
KB	Knowledge Base
MS	Milestone
NFR	Non-Functional Requirements
ORSD	Ontology Requirements Specification Document
OWL	Web Ontology Language
PUC	Pilot Use Case
RDF	Resource Definition Language
SEM	Simple Event Model
SPARQL	SPARQL Protocol and RDF Query Language
SPIN	SPARQL Inferencing Notation
SWRL	Semantic Web Rule Language
SWRL	Semantic Web Rule Language
VR	Virtual Reality
WP	Work Package

Table of Contents

1 INTRODUCTION

One of the cardinal objectives of WP5 is to provide the framework for encoding, aggregating (T5.1), and semantically analysing information (T5.2) relevant to the V4Design application domain. In particular, WP5 provides the knowledge structures and vocabularies (ontologies) for capturing the structure and semantics of:

- Aesthetic concept extraction for emotion recognition from images (WP3)
- Key concepts, named entities and conceptual relations extracted via/through the analysis and understanding of textual content (WP3)
- Localization of the interior and exterior of buildings in visual content that determine the presence of buildings and interior objects in movies and documentaries (WP4)
- 3D model reconstruction metadata derived from the visual content (WP4)
- Relations among 3D objects with their sub-elements of the constructed BIM or GIS models (WP5 / T5.3)
- Text generation that summarises basic attributes of the generated 3D objects (WP5 / T5.4, T5.5)

The logical dependencies of WP5 with the other WPs of the V4Design project are depicted in Figure 1. The Figure also depicts the dependencies with WP6 and WP7 relevant to the development of the modules that will be integrated in the system and the feedback needed from the users with respect to requirements and evaluation.

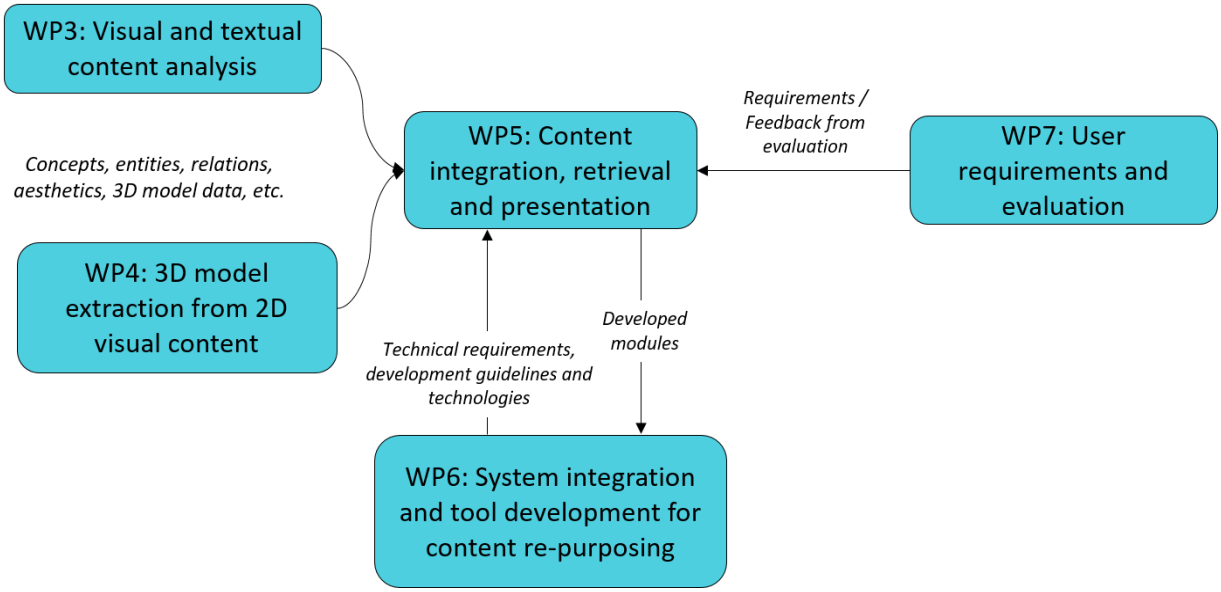

Figure 1: Logical dependencies of WP5 with the other WPs

In order to promote interoperability, extensibility and sharing, WP5 reuses and extends existing standards for defining the vocabulary of the annotations, as well as the patterns for associating these vocabularies with the generated assets. More specifically, the metadata vocabularies are defined in the Web Ontology Language (OWL 2[1] (Grau et al. 2008)), the W3C standard for defining and sharing ontologies. Similarly, the metadata are associated with assets using the Web Annotation Data Model (Sanderson, Ciccarese, and Young 2017),

[1] https://www.w3.org/TR/owl-profiles/

which provides an extensible and interoperable framework for expressing annotations. This model was published by the W3C Web Annotation Working Group as a Recommendation (since 23 February 2017), describing a common approach to express annotations in a manner that is simple and convenient, while at the same time enables more complex requirements.

The population of the V4Design ontologies is done automatically by mapping the information provided as input by other components of the system. To this end, WP5 develops the necessary algorithms and interfaces for the structural and semantic mapping of data among different schemas and vocabularies, creating interlinked RDF-based knowledge structures pertinent to the assets derived by the V4Design modules. For example, the aesthetics and objects extracted from different modules on the same visual content are interlinked in the WP5, so as to create a unified metadata annotation graph.

Finally, WP5 provides the reasoning layer, whose purpose is to address WP5's reasoning requirements, in the form of ontology-based axiomatisations (e.g. complex class descriptions and property axioms) and inference rules. The underlying reasoning techniques will afford the derivation of knowledge-driven interpretations, enabling the system to abstract from incoming information and enrich the underlying knowledge graphs. This combination of semantically rich and interlinked knowledge graphs will foster the retrieval of assets based on semantic relationships and not simply on keyword-based search, improving the understanding of the users' intent and the contextual meaning of the provided terms. In addition, the interpretation framework will be able to cope with partial and imperfect information, recognising the context of the incoming information that will be semantically coupled and interlinked with semantic knowledge structures.

Figure 2 presents the conceptual architecture of WP5 that consists of the following entities:

- Knowledge Base (KB), that provides native RDF storage and querying services
- Population, which implements the mapping services of input provided by other components
- Reasoning, which implements the reasoning framework combining native OWL 2 reasoning and custom rules
- Linked Data, which implements linked data design principles to further enrich the derived 3D models
- Text Generation, which generates descriptive text based on the metadata of the assets in the KB

The remainder of this document is structured as follows: Section 2 overviews the adopted methodology for the creation of the V4Design ontologies. Section 3 describes the user requirements that are relevant to WP5 modelling and reasoning framework. Section 4 reports the modelling specifications of the V4Design ontologies. Section 5 reviews the relevant literature for reasoning, ontologies and annotation models. Section 6 presents the first version of the V4Design ontologies guided by the specifications (section 4), along with the modelling insights derived from the literature analysis (section 5). It also describes the basic principles that underpin the first preliminary version of WP5's reasoning framework towards intelligent knowledge-driven reasoning, contextualised aggregation and interpretation. Section 7 presents how the created models are applied on a set of examples for validation purposes, while Section 8 discusses the results and concludes the document.

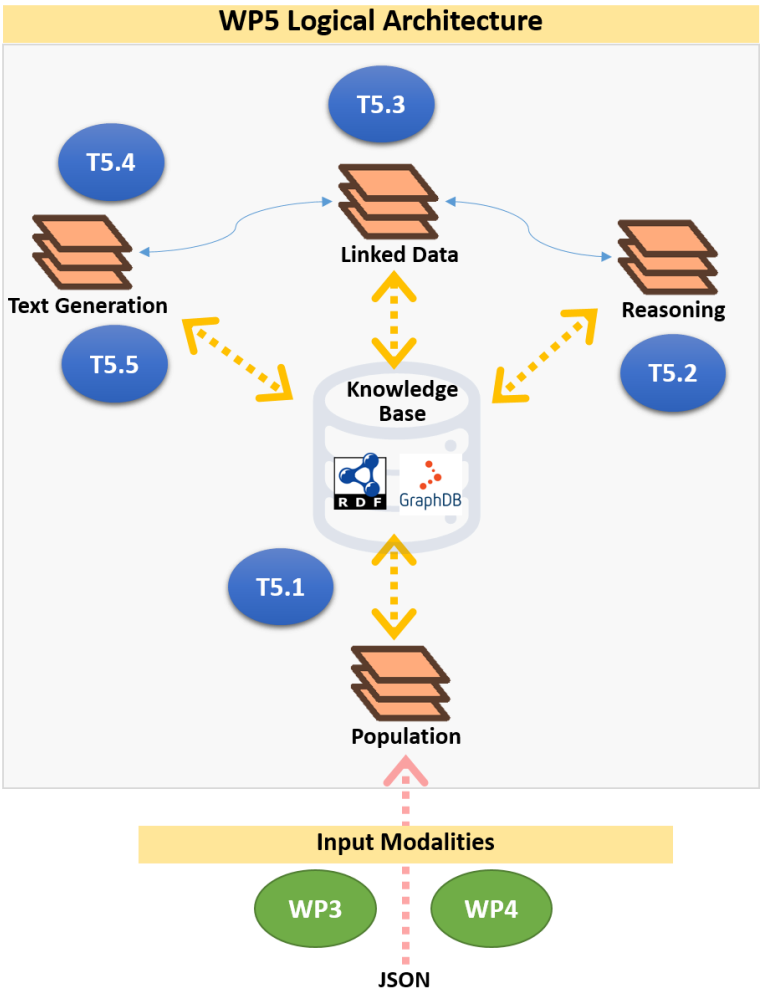

Figure 2: Conceptual architecture of the modules involved in WP5

2 METHODOLOGY FOR MODELLING REQUIREMENTS

2.1 Ontology development 101 methodology

There are many ways to model a domain using ontologies and the ontology development is essentially an iterative process. In this sense, there are several methodologies for ontological engineering such as On-To-Knowledge (OTK) (Staab et al. 2001), METHONTOLOGY (Fernández-López, Gómez-Pérez, and Juristo 1997), United Process for Ontologies (UPON) (De Nicola, Missikoff, and Navigli 2005) and Ontology Development 101 (Noy and McGuinness 2001). Most of these methodologies introduce common features and development guidelines.

For the purposes of the V4Design ontological framework, we adopted the methodology of *Ontology Development 101* which consists of the following iterative steps:

Step 1. Determination of the domain and scope of the ontology
Step 2. Reuse of existing ontologies
Step 3. Enumeration of important terms
Step 4. Definition of the classes and the class hierarchy
Step 5. Definition of the properties
Step 6. Creation of instances.

In literature, the determination of the domain and scope of the ontology can be documented in a template-based report called “Ontology Requirements Specification Document” (ORSD) (Suárez-Figueroa, Gómez-Pérez, and Villazón-Terrazas 2009). This document allows the systematic specification of “why the ontology is being built”, “what its intended uses are”, “who the end-users are”, and “which requirements the ontology should fulfil”. In particular, the ORSD report contains the following fields:

1. **Purpose**: the main general goal of the ontology (i.e. how the ontology will be used in V4Design)
2. **Scope**: the general coverage and the degree of detail of the ontology
3. **Implementation language**: the formal language of the ontology
4. **Intended end-users**: the intended end-users expected for the ontology
5. **Intended uses**: the intended uses expected for the ontology
6. **Ontology requirements**
    a. **Non-functional requirements**: the general requirements or aspects that the ontology should fulfil, including optional properties for each requirement
    b. **Functional requirements**: the content specific requirements that the ontology should fulfil in the form of groups of competency questions and their answers, including optional priorities for each group and for each competency questions
7. **Pre- Glossary of terms**
    a. **Terms from competency questions**: the list of items included in the competency questions and their frequencies
    b. **Terms from answers**: the list of terms included in the answers and their frequencies
    c. **Objects**: the list of objects included in the competency questions and their answers

Before presenting the V4Design ORSD (section 4), we outline the WP5 relevant application context within which the V4Design ontology is deployed (section 3).

3 USER REQUIREMENTS RELEVANT TO ONTOLOGIES & REASONING

This section presents the application context relevant to WP5, describing relevant user requirements that drive the development of the V4Design modelling and reasoning framework. To this end, we have investigated the description of the context and the users for each scenario, as well as the requirements that have been presented in ***"D7.2: Use cases, requirements and evaluation plan".*** These requirements will be translated into technical requirement in ***"D6.2: Technical requirements and architecture".*** Table 1 presents the user requirements that are relevant to the WP5 representation and reasoning framework, briefly describing the main functionalities and services that need to be supported.

User Requirement ID (D7.2)	Description	WP5 Relevance / Dependency
UR_2	As an Architect I want to be able to retrieve 3D-Models	• Provide the annotation models (ontologies) to represent and integrate analysis results from other modules • Support searching functionality and interface over the KB with the collected metadata
UR_3	As an Architect I want to be able to retrieve high and reduced resolution textures	• Capture metadata about the texture resolution
UR_10	As a user I want further details about the acquired footage - image/ video (semantic data/ tags)	• Capture metadata and tags coming from building and object localisation, aesthetics, text analysis, reasoning
UR_12	As a user I want further details about the extracted data quality	• Capture metadata about quality of assets
UR_15	As a user I want further details about geo-location and date/ time of scan	• Support the annotation of geo-location and date information
UR_16	As a user I want further details about the author and copyrights of the asset	• Support the annotation of authors and copyright info
UR_18	As a user I want further details about visible colours in the asset	• Support the annotation of visible colours
UR_20	As a user I want augmented data of the acquired 3D model (semantic data/ tags)	• Capture metadata and tags coming from building and object localisation, aesthetics, text analysis, reasoning

UR_21	As a user I want a description of the acquired 3D model	• Support the annotation of assets with results from text generation
UR_22	As a user I want related Wikipedia articles to the acquired 3D model	• Ability to associate assets with relevant external Web Pages
UR_23	As a user I want summarizations of textual content related to the 3D model	• Support the annotation of assets with results from text generation
UR_30	As an Architect I want UIX: 3D-gallery i.e. a distraction free interface with rendered preview thumbnails	• Ability to associate assets with preview thumbnails
UR_35	As an Architect I want UIX: Search by semantic tags (keywords)	• Support search functionality and interface over the KB with the collected metadata
UR_37	As an Architect I want UIX: Detailed search by features: - Quality (3D model/ texture), Footage features, augmented data	• Support the annotation of assets with relevant metadata
UR_41	As an Architect I want texture and material recognition that might appear in images and videos.	• Ability to annotate textures and materials
UR_50	As a user I would like to have access to lists of 3D models, but also find contextual information, other assets and related work	• Support searching functionality and interface over the KB with the collected metadata • Support the linking of relevant assets
UR_55	As a content provider I want clear and transparent labelling of the reuse rights and copyright status of every item in the V4Design platform so as to enable better communication and IP protection to content providers.	• Support the annotation of assets with reuse rights and copyright information
UR_57	As a game designer I want to get information about the assets - Textual and semantic data about the 3D assets - Textual summaries describing the 3D models	• Support the annotation of assets with textual descriptions and summaries

Table 1: User requirements relevant to WP5 representation and reasoning framework

4 ONTOLOGY REQUIREMENT SPECIFICATION DOCUMENT

This section presents the ORSD which provides the specification of the V4Design ontological framework. The ORSD may be further elaborated and extended as the system functionalities will evolve to take into account hidden and unrevealed aspects that are not covered by the current ontology requirements.

V4Design ORSD	
1	**Purpose**
	The purpose of the V4Design representation framework is to provide the ontological structures and vocabularies (ontologies) to capture the results of the V4Design analysis modules in a reusable and interoperable manner. To this end, the ontological framework will provide the annotation model needed in order to support data modelling, integration and reasoning over the distilled information. These include: • Constructs for capturing metadata of different resources, such as aesthetics, recognised buildings and objects, named entities and concepts, etc. • A structured model and format to enable annotations and assertions to be defined, shared and reused across both within the V4Design application context but also within different hardware and software platforms.
2	**Scope**
	The V4Design ontology has to formally capture: • Aesthetics extraction-related information derived from the analysis of visual content • Building/object localisation-related information derived from the spatio-temporal analysis of visual content • 3D reconstruction-related information derived from the multiple-view reconstruction analysis of visual content • Object-related information derived by the further enrichment of 3D models with additional linked data towards textual and visual analysis • Textual-related annotations that are derived by text analysis and generation. A key design choice underpinning the engineering of the V4Design ontologies is the adherence to a pattern-based approach, so as to capitalise on a modular, extensible and interoperable framework for expressing annotations and achieve a better degree of knowledge sharing, reuse and interoperability.
3	**Implementation Language**
	The ontology will be implemented in OWL 2 (Grau et al. 2008), the officially recommended language by W3C for knowledge representation in the Semantic Web.
4	**Intended End-Users**

	The V4Design system considers different types of end users depending on the application context, who will interact with the generated knowledge through authoring tools: • ***PUC1: Architectural design, related to existing or historical buildings and sites and their environments*** **Architects and designers:** Architects and designers who want to reconstruct (design/refurbish/extend) the surrounding landscape and the various spatial elements that articulate it. • ***PUC2: Architectural design, related to artworks, historic or stylistic elements*** **Architects, designers and artists:** Architects, designers and artists who want to reinterpret key aspects of artworks and produce designs and objects (e.g. furniture collections, decorative objects, lighting accessories, etc.) that are original but at the same time stylistic and historically charged. • ***PUC3: Design of virtual environments, related to TV series and VR video games architectural design, artworks, historic or stylistic elements*** **Content providers:** Content providers who want to provide a more immersive experience to its audiences by developing immersive environments and 3D assets of the objects from their video archives. • ***PUC4: Design of virtual environments related to actual news for VR (re) living the date*** **Content providers:** Content providers who want to develop interactive and immersive documentaries using the existing footage they have from various news and locations.
5	**Ontology Requirements**
	Non-functional requirements
	NFR1. The ontology should adopt available standards whenever possible and reuse existing ontologies and vocabularies
	Functional requirements: Groups of competency questions
	The list of Competency Questions (CQ) below has been derived by studying the Pilot Use Case scenarios and user requirements. The questions have been also elicited through the direct interaction with technical partners. To this end, a simulation example has been carried out in order to collect additional technical and user-related requirements that drive the development of the annotation models (see Section 7). As described in the Data Management Plan (D1.2), there are different categories of data. Similarly, the list of CQ contains questions which correspond to these different types of data. **Visual content** 3D Models

	CQ1 Which is the identifier of the 3D model? **CQ2** Which is the title of the 3D model? **CQ3** Which is the description and/or summary of the 3D model? **CQ4** Which is the creation date of the 3D model? **CQ5** Who is the creator of the 3D model? **CQ6** Which is the historic period/style of the asset depicted by the 3D object? **CQ7** Which is the artist of the asset depicted by the 3D object? **CQ8** Which is the object type of the asset depicted by the 3D object? **CQ9** Which is the building type of the asset depicted by the 3D object? **CQ10** Which is the material/texture of the asset depicted by the 3D object (e.g., marble, stone, brown, seamless, architecture)? **CQ11** Which is the construction date of the asset depicted by the 3D model? **CQ12** Which is the location (e.g., coordinates) of the asset depicted by the 3D model? **CQ13** Which is the images from which the 3D model has been reconstructed? **CQ14** Which is the point cloud associated with this 3D model? **CQ15** Which is the identifier of the thumbnail of the point cloud? **CQ16** Which is the mesh associated with this 3D model? **CQ17** Which is the identifier of the thumbnail of the 3D reconstructed mesh? **CQ18** Which is the licence of the visual content from which the 3D model has been reconstructed? **CQ19** Which are the 3D models derived from the same visual content? **CQ20** Which are the relevant 3D models in terms of artist, period/style, texture, building/object type? **CQ21** Which textual content (e.g., posts, captions/descriptions) is related to this 3D model? **CQ22** For what kind of application is the 3D model suitable (e.g. low performance mobile applications or high end productions)? **CQ23** For what kind of application is the mesh suitable? **CQ24** Is this 3D model suitable for a top view mobile or a real size virtual environment? **CQ25** What is the file size of the 3D model? **CQ26** What is the source format of the 3D model (e.g., .obj, .fbx)? **CQ27** Which are the dimensions (real size) and the scale of the original asset depicted by the 3D model? **CQ28** What is the thumbnail file size of the point cloud? **CQ29** What is the thumbnail file size of the 3D reconstructed mesh? **CQ30** Which is the number of points for the point cloud associated to this 3D model? **CQ31** Which is the number of quads of the mesh associated to this 3D model? **CQ32** Which is the number of triangles of the mesh associated to this 3D model? **CQ33** Which is the number of polygons of the mesh associated to this 3D model? **CQ34** Which is the number of converted polygons to triangles of the mesh associated to this 3D model? **CQ35** Which is the number of vertices of the mesh associated to this 3D model?

	Textures **CQ36** Which is the identifier of the texture? **CQ37** Which is the material of the texture? **CQ38** Which is the map of the texture? Video from V4Design content providers and external sources **CQ39** Which is the identifier of the video? **CQ40** Which is the title of the video? **CQ41** Which is the description of the video? **CQ42** Which is the URL of the video? **CQ43** Which the URL of the thumbnail of the video? **CQ44** Which is the provider/creator/producer of the video? **CQ45** Which is the Web resource from which the video is retrieved? **CQ46** Which are the tags assigned to the video? **CQ47** Which is the EDM ID associated with the specific video? **CQ48** Which is the licence of the video? **CQ49** What is the file size of the video? **CQ50** What is the file type of the video? **CQ51** What is the video length (e.g., 90 seconds)? **CQ52** How many frames per second does the video have? **CQ53** What are the video dimensions (e.g. 720 pixels x 576 pixels)? **CQ54** What is bit-rate (kbps) of the video? **CQ55** Which is the file with associated metadata? Images from V4Design content providers and external sources (Flickr, Wikipedia, etc.) **CQ56** Which is the identifier of the image? **CQ57** Which is the title of the image? **CQ58** Which is the caption/description of the image? **CQ59** Which is the URL of the image? **CQ60** Which the URL of the thumbnail of the image? **CQ61** Which is the provider/creator/producer of the image? **CQ62** Which is the Web resource from which the image is retrieved? **CQ63** Which are the tags assigned to the image? **CQ64** Which is the EDM ID associated with the specific image? **CQ65** Which is the licence of the image? **CQ66** What is the file size of the image? **CQ67** What is the file type of the image? **CQ68** What is the resolution of the image? **Textual Content** Captions/descriptions associated with a visual content **CQ69** Which is the identifier of the caption/description? **CQ70** Which is the visual content (image/video) associated with the specific caption/description?

	CQ71 Which is the Web resource from which the caption/description is retrieved? **CQ72** Which is the licence of this textual content? Textual descriptions not directly associated with a specific visual content (e.g., Wikipedia pages, news and magazine articles) **CQ73** Which is the identifier of the text? **CQ74** Which is the Web resource from which the text is retrieved? **CQ75** Which is the licence of this textual content? **CQ76** Which is the visual content (image/video) associated with the specific caption/description? **Aesthetics** **CQ77** In which image did we perform aesthetics extraction? **CQ78** Who is the artist of the depicted artwork in the image? **CQ79** What is the style/period of the depicted artwork in the image? **Building/Object Localisation** **CQ80** What is the id of the mask? **CQ81** What is the file size of the created mask? **CQ82** What is the file type of the created mask (.jpeg, .png, etc.)? **CQ83** What is the resolution of the created mask? ***Building localisation*** **CQ84** In which image did we perform building localisation? **CQ85** In which part of the image did we locate a building? **CQ86** What type of building is depicted in the specific part of the image? **CQ87** What is the mark we crated based on this building? ***Object localisation*** **CQ88** In which image did we perform object localisation? **CQ89** In which part of the image did we locate an object? **CQ90** What type of object is depicted in the specific part of the image? **CQ91** What is the mark we created based on this object? **Metadata Annotations** **CQ92** Which component provided the annotations? **CQ93** What is the timestamp of the annotations?

The competency questions cover a very exhaustive list of aspects relevant to the V4Design domain. Though the initial set of V4Design ontologies will not cover this extended list and depth of detail, they provide the modular structures that will enable the future extensibility of the model.

5 STATE OF THE ART

This section provides an overview on the relevant state of the art with respect to knowledge representation languages as well as already existing ontologies addressing project-relevant fields. More specifically, we present the basics of Description Logic (DL) languages (Baader et al. 2003), on which the official W3C recommendation for creating and sharing ontologies in the Web (OWL 2) is grounded, the different OWL 2 species, as well as relevant rule-based languages. We then provide a briefly review on the representative ontologies that have been proposed in the literature for modelling core aspects relevant to the V4Design application domain that fall into WP5's modelling requirements.

5.1 Web Ontology Language

In the literature, ontologies have been widely used as an effective way for modelling domain information because they can represent and organize information, context and relationships more accurately. In addition, they offer easy expandability by merging, expanding and combining parts of existing ontologies into new ones.

Ontologies are models used to capture knowledge about some domain of interest. Formally speaking, ontologies are *explicit formal specifications of shared conceptualizations* (Gruber 1993; Studer, Benjamins, and Fensel 1998). They represent abstract views of the world including the objects, concepts, and other entities that are assumed to exist in some area of interest, their properties and the relationships that hold among them. Their expressivity and level of formalisation depend on the knowledge representation language used.

Within the Semantic Web, which is an extension of the current Web that aims to establish a common framework for sharing and reusing data across heterogeneous sources, ontologies play a key role. The Semantic Web vision is to make the semantics of web resources explicit by attaching to them metadata that describe meaning in a formal, machine-understandable way. In this effort, the Web Ontology Language (OWL) (Deborah L. McGuinness 2004) has emerged as the official W3C recommendation for creating and sharing ontologies on the Web. In the rest of this section, we present the basics of Description Logic (DL) languages, on which OWL semantics are grounded, the different OWL species, as well as relevant rule-based languages.

5.1.1 DL Reasoning

Description Logics (DLs) (Baader et al. 2003) are a family of knowledge representation formalisms characterised by logically grounded semantics and well-defined reasoning services. The main building blocks are *concepts* representing sets of objects (e.g. `Person`), *roles* representing relationships between objects (e.g. `worksIn`), and *individuals* representing specific objects (e.g., `Alice`). Starting from *atomic* concepts, such as `Person`, arbitrary complex concepts can be described through a rich set of *constructors* that define the conditions on concept membership. For example, the concept $\exists$`hasFriend.Person` describes those objects that are related through the `hasFriend` role with an object from the concept `Person`; intuitively, this corresponds to all those individuals that are friends with at least one person. A DL knowledge base K typically consists of a $TBox\ T$ (terminological knowledge) and an $ABox\ A$ (assertional knowledge). The TBox contains axioms that capture the possible ways in which objects of a domain can be associated. For example, the TBox

axiom `Dog ⊑ Animal` asserts that all objects that belong to the concept `Dog`, are members of the concept `Animal` too. The ABox contains axioms that describe the real world entities through concept and role assertions. For example, `Dog(Jack)` and `isLocated(Jack,kitchen)` express that `Jack` is a dog and he is located in the kitchen. Table 2 summarises the set of terminological and assertional axioms.

Name	Syntax	Semantics
Concept inclusion	$C \sqsubseteq D$	$C^{\mathrm{I}} \subseteq D^{\mathrm{I}}$
Concept equality	$C \equiv D$	$C^{\mathrm{I}} = D^{\mathrm{I}}$
Role Equality	$R \equiv S$	$R^{\mathrm{I}} = S^{\mathrm{I}}$
Role inclusion	$R \sqsubseteq S$	$R^{\mathrm{I}} \subseteq S^{\mathrm{I}}$
Concept assertion	$C(\alpha)$	$\alpha^{\mathrm{I}} \in C^{\mathrm{I}}$
Role assertion	$R(\alpha,b)$	$(\alpha^{\mathrm{I}}, b^{\mathrm{I}}) \in R^{\mathrm{I}}$

Table 2: Terminological and assertional axioms

5.1.2 **DL reasoning services**

DLs come with a set of powerful reasoning services, for which efficient, sound and complete reasoning algorithms with well understood computational properties are available. Example state-of-the-art implementations include Pellet (Sirin et al. 2007) Racer (Haarslev and Möller 2003), Fact++ (Tsarkov and Horrocks 2006) and Hermit (Glimm et al. 2014).

Assuming a DL knowledge base $K = (T, A)$, typical reasoning services include:

- **Subsumption:** A concept *C* is subsumed by *D* in *T* (written $T \models C \sqsubseteq D$), *iff* $C^{\mathrm{I}} \subseteq D^{\mathrm{I}}$ for all interpretations I.
- **Equivalence:** Two concepts *C* and *D* are equivalent in *T* (written $T \models C \equiv D$) *iff* $C^{\mathrm{I}} \subseteq D^{\mathrm{I}}$ and $D^{\mathrm{I}} \subseteq C^{\mathrm{I}}$ *for all interpretations* I.
- **Disjoint:** A concept *C* is disjoint to a concept *D* in *T iff* in every interpretation I it holds that $C^{\mathrm{I}} \neq \emptyset$.
- **Consistency:** The ABox *A* is consistent w.r.t. *T iff* if there is an interpretation that is a model of both *A* and *T*.
- **Instance checking:** The individual α is an instance of *C* (w.r.t. *K*) (written $K \models C(\alpha)$) *iff* $\alpha^{\mathrm{I}} \in C^{\mathrm{I}}$ holds for all interpretations I of *K*.
- **Realisation:** The realisation of an instance α w.r.t. to *K* includes finding the most specific concepts *C* for which $a^{\mathrm{I}} \in C^{\mathrm{I}}$ holds for all interpretations I of *K*.

Hence, through subsumption one can derive the implicit taxonomic relations among the concepts of a terminology. For example, given the axiom `OccupiedRoom ≡ Room ⊓ ∃contains.Person`, one can infer that `Room` subsumes `OccupiedRoom`.

Satisfiability and consistency checking are useful to determine whether a knowledge base is meaningful at all. Satisfiability checking enables the identification of concepts for which it is impossible to have members under any interpretation (for example, an unsatisfiable concept, though trivial, is `OccupiedRoom ⊓ ¬OccupiedRoom`). Consistency checking enables the identification whether the set of assertions comprising the knowledge base is admissible with respect to the terminological axioms. For example if `EmptyRoom` and

`OccupiedRoom` are asserted as disjoint concepts, then the presence of both `OccupiedRoom(kitchen)` and `EmptyRoom(kitchen)` leads to inconsistency.

Instance checking denotes the task of finding whether a specific individual is an instance of a given concept. Realisation of an individual, a more generic form of instance checking, returns all (most specific) concepts from the knowledge base that a given individual is an instance of. Its dual is the retrieval problem that given a specific concept *C*, it returns all individuals that belong to this concept. This reasoning service is the central to realise the task of recognition of situation types.

Falling under the classical logics paradigm, reasoning in DLs adopts the open-world assumption. Intuitively, if a fact α holds only in a subset of the models of the knowledge base *KB*, then we can conclude neither $KB \models \alpha$ nor $KB \models \neg\alpha$. For example, if the only available knowledge regarding the residents of a house is the assertion `livesIn(Alice,house)`, we cannot deduce based on it alone that no one else lives in the house. In contrast, formalisms adhering to the closed-world assumption make the common-sense conjecture that all relevant information is explicitly known, so all unprovable facts should be assumed not to hold. In our example, this amounts to concluding that Alice is the sole resident of this house.

Hence, closed-world reasoning can be intuitively understood as reasoning where from $KB \models \alpha$, one concludes $KB \models \neg\alpha$. Such kind reasoning should not be confused however with closed domain reasoning, which involves reasoning only over explicitly known individuals.

5.1.3 **OWL and OWL 2**

The OWL is a knowledge representation language widely used within the Semantic Web community for creating ontologies. The design of OWL and particularly the formalisation of the semantics and the choice of language constructors have been strongly influenced by DLs. OWL comes in three dialects of increasing expressive power: OWL Lite, OWL DL and OWL Full. OWF Full is the most expressive of the three: it neither imposes any constraints on the use of OWL constructs, nor lifts the distinction between instances (individuals), properties (roles) and classes (concepts). This high degree of expressiveness comes however at a price, namely the loss of decidability that makes the language difficult to implement. As a result, focus has been placed on the two decidable dialects, and particularly on OWL DL, which is the more expressive of the two.

Despite the rich primitives provided for expressing concepts, OWL DL has often proven insufficient to address the needs of practical applications. This limitation amounts to the DLs style model theory used to formalise its semantics, and particularly the *tree model property* (Vardi 1996) conditioning DLs decidability. As a consequence, OWL can model only domains where objects are connected in a tree-like manner. This constraint is quite restrictive for many real-world applications, including the ambient intelligence domain, which requires modelling general relational structures.

Responding to this limitation and to other drawbacks that have been identified concerning the use of OWL in different application contexts throughout the years, the W3C working group produced OWL 2 (Grau et al. 2008). OWL 2 is a revised extension of OWL, now commonly referred to as OWL 1. It extends OWL 1 with qualified cardinality restrictions; hence one can assert for example that a social activity is an activity that has more than one actor: $\texttt{SocialActivity} \equiv \texttt{Activity} \sqcap \geq 2\texttt{hasParticipant.Person}$.

Another prominent OWL 2 feature is the extended relational expressivity that is provided through the introduction of complex property inclusion axioms (property chains). To maintain decidability, a regularity restriction is imposed on such axioms that disallow the definition of properties in a cyclic way. Hence, one can assert the inclusion axiom `locatedIn` ∘ `containedIn` ⊑ `locatedIn` making it possible to infer that if a person is located for example in the bedroom of her house, then she is located in her house as well; however, it is not allowed to use both the aforementioned axiom and the axiom `containedIn` ∘ `locatedIn` ⊑ `containedIn` as this leads to a cyclic dependency. Three profiles, namely OWL 2 EL, OWL 2 QL and OWL 2 RL, trade portions of expressive power for efficiency of reasoning targeting different application scenarios.

5.1.4 **Rules**

To achieve decidability, DLs, and hence OWL, trade some expressiveness for efficiency of reasoning. The tree-model property is one such example. It conditions the tree-shape structure of models, ensuring decidability, but at the same time it severely restricts the way variables and quantifiers can be used, dictating that a quantified variable must occur in a property predicate along with the free variable. As a result, it is not possible to describe classes whose instances are related to an anonymous individual through different property paths. To leverage OWL's limited relational expressivity and to overcome modelling shortcomings that OWL alone would be insufficient to address, a significant body of research has been devoted to the integration of OWL with rules.

A proposal towards this direction is the Semantic Web Rule Language (SWRL) (Horrocks et al. 2004), in which rules are interpreted under the classical first order logic semantics. Allowing concept and role predicates to occur in the head and the body of a rule without any restrictions, SWRL maximises the interaction between the OWL and rule components, but at the same time renders the combination undecidable. To regain decidability, several proposals have explored syntactic restrictions on rules (Motik, Sattler, and Studer 2005; Rosati 2006) as well as their expressive intersection of Description Logic Programs (DLP) (Grosof et al. 2003). The DL-safe rules introduced for example in (Motik, Sattler, and Studer 2005) impose that rule semantics apply only over known individuals. It is worth noting that in practice DL reasoners providing support for SWRL actually implement a subset of SWRL based on this notion of DL-safety.

Taking a different perspective, a number of approaches have investigated the combination of ontologies and rules based on mappings of a subset of the ontology semantics on rule engines. For instance, (ter Horst 2004) defines the *pD** semantics as a weakened variant of OWL Full, e.g., classes can be also instances, and they are extended to apply to a larger subset of the OWL vocabulary, using 23 entailments and 2 inconsistency rules. Inspired by the *pD** entailments and DLP, the semantics of the OWL 2 RL profile is realised as a partial axiomatisation of the OWL 2 semantics in the form of first-order implications, known as OWL 2 RL/RDF rules. User-defined rules on top of the ontology allow expressing richer semantic relations that lie beyond OWL's expressive capabilities, and couple ontological and rule knowledge.

SPARQL (Harris and Seaborne 2013) is a declarative language recommended by the W3C for extracting and updating information in RDF graphs. It is an expressive language that allows the description of quite complex relations among entities. The semantics and complexity of

the SPARQL query language have been fairly studied theoretically, showing that SPARQL algebra has the same expressive power as relational algebra (Perez, Arenas, and Gutierrez 2006) (He et al. 2004). Although SPARQL is mostly known as a query language for RDF, by using the `CONSTRUCT` graph pattern, it is able to define SPARQL rules that can create new RDF data, combining existing RDF graphs into larger ones. Such rules are defined in the interpretation layer in terms of a `CONSTRUCT` and a `WHERE` clause: the former defines the graph patterns, i.e. the set of triple patterns that should be added to the underlying RDF graph upon the successful pattern matching of the graphs in the `WHERE` clause. The SPARQL Inferencing Notation (SPIN) (Knublauch, Hendler, and Idehen 2011) constitutes an effort to ease the definition and execution of SPARQL rules on top of RDF graphs. In SPIN, SPARQL queries can be stored as RDF triples together with any RDF domain model, enabling the linkage of RDF resources with the associated SPARQL queries, as well as sharing and reuse of SPARQL queries. SPIN supports the definition of SPARQL inference rules that can be used to derive new RDF statements from existing ones through iterative rule application.

5.2 Ontologies relevant to the V4Design domain

This section briefly reviews state-of-the-art ontologies that can be used for modelling core aspects relevant to the V4Design application domain. According to the V4Design ontological requirements, we have spit the relevant ontologies into four sections. First, we review ontologies that can be used to model events and observations (in the sense that V4Design module outputs can be considered as events), then we continue with general purpose ontologies that provide the conceptual background for modelling generic content and context. We then present ontologies that have been mainly used to capture multimedia information and finally existing patterns that have been proposed to define annotation models.

It should be noted that the purpose of this section is not to provide a complete list of ontologies relevant to the V4Design domain, but to elaborate on design principles that have been followed in the literature for defining annotations and conceptual models.

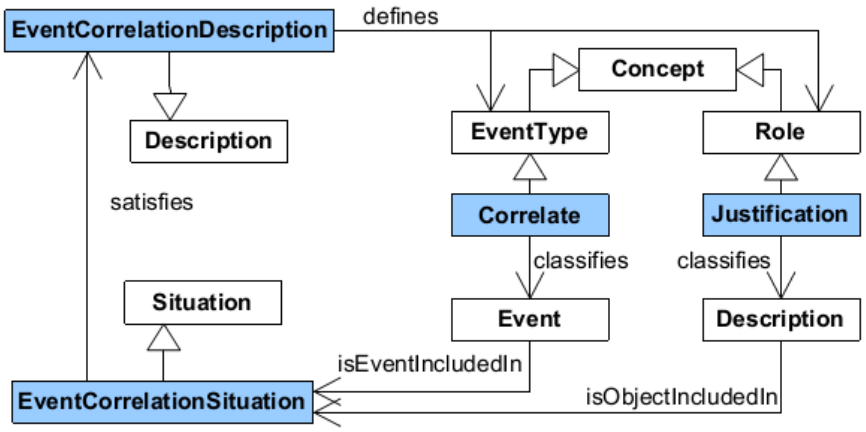

Figure 3: The event correlation pattern in Event-Mode-F

5.2.1 Observations and events

Event-Model-F

Event-Model-F (Scherp et al. 2009) defines an expressive model for capturing and representing occurrences in the real world. It is based on DUL, following the descriptions and situations ontology design pattern (DnS) (Gangemi and Mika 2003) for modelling aspects of events, such as object participation, mereological, causal, and correlative relationships, and different interpretations of the same event (by reifying events in order to describe n-ary relations) introducing six ontology design patterns.

DnS enhances DOLCE's descriptive characteristics even further allowing the context-sensitive "redescriptions" of the types and relations postulated by other given ontologies or ground vocabularies. The current OWL encoding of DnS assumes DOLCE as a ground top-level vocabulary. The basic implementation of the DnS pattern in DUL allows the relation of situations (`dul:Situation`) and descriptions (`dul:Description`) with domain events (`dul:Event`), concepts (`dul:EventType`), objects and participants. More specifically, a situation describes the entities of a context, e.g. the events and objects that are involved, and satisfies (`dul:satisfies`) a description. The description in turn defines (`dul:defines`) concepts that classify (`dul:classifies`) the entities of the situation, describing the way they should be interpreted. Event-Model-F implements a number of instantiations on top of the DnS pattern to describe relations among events, such as causality and correlation. For example, Event-Model-F (`emf` namespace) allows the association of composite events with their sub-events through descriptions that use the concepts `emf:Composite` ⊑ `dul:EventType` and `emf:Component` ⊑ `dul:EventType`. Figure 3 depicts the event correlation pattern of Event-Model-F.

Simple Event Model

The Simple Event Model (SEM) (Van Hage et al. 2011) is an effort to define an ontology model for events without strong semantic constraints. This decision is justified by the open

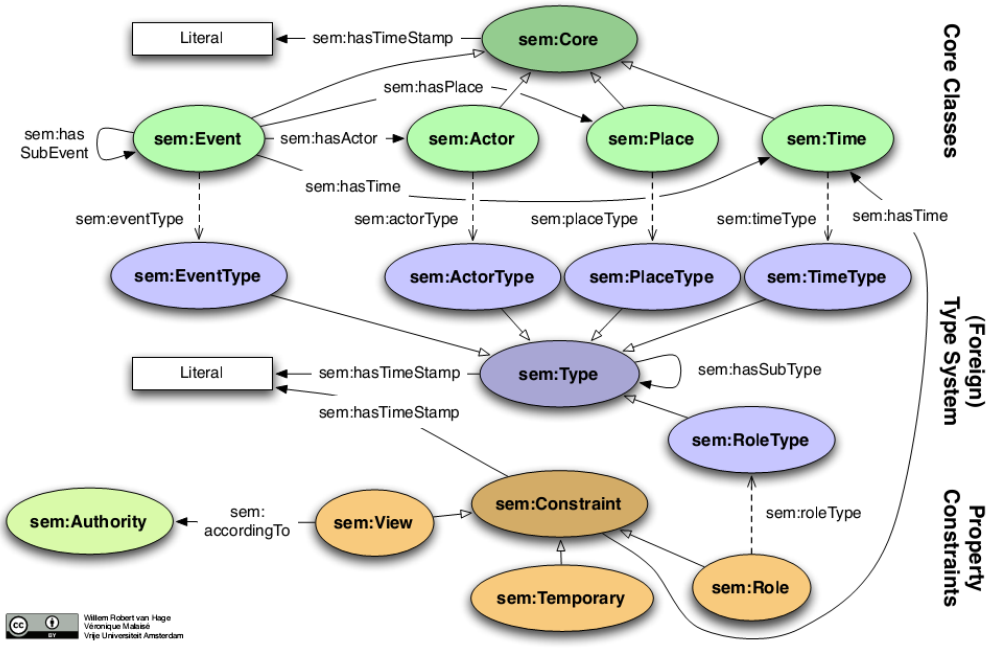

Figure 4: Simple Event Model

nature of the Web and the need to model different (even conflicting) views of the same event. The lack, however, of strong semantic constraints, such as functional properties, disjoint classes and cardinality restrictions, hampers the ability to automatically validate and resolve model inconsistencies using formal inference mechanisms. Therefore, SEM is characterised by a trade-off between model reusability and automated reasoning and validation capabilities. Figure 4 presents the main concepts of this ontology.

5.2.2 Content and context

PROV Ontology (PROV-O)

The PROV Ontology (PROV-O) (Lebo, Sahoo, and McGuinness 2013) provides a set of classes, properties, and restrictions to represent the provenance information associated with data published (Figure 5). The core concepts of the conceptual model revolves around the notion of entity, a digital, physical or other thing; activity, an action generating or manipulating entities; and agent, the responsible person/institution/administration for an activity taking place as it did. The PROV Ontology classes and properties are defined such that not only they can be used directly to represent provenance information, but they can also be specialized for modelling application-specific provenance details in a variety of domains.

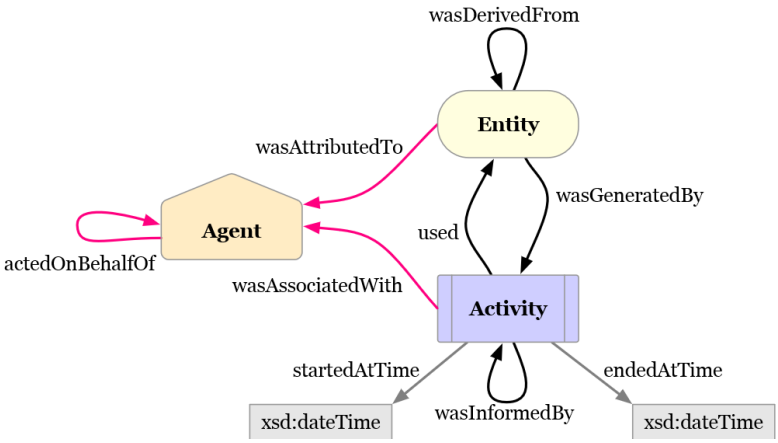

Figure 5: The core concepts of the PROV ontology

PAV (Provenance, Authoring and Versioning)

PAV (Ciccarese et al. 2013) extends the PROV-O and specifies Provenance, Authoring and Versioning information. Compared to PROV-O, it focuses on the provenance of a digital resource in terms of its relationships with other digital resources and agents involved in their creation, authoring and manipulation, and it abstracts away from the description of the activities (process) that manipulate and transform the digital resources.

More specifically, the PAV ontology (Figure 6) provides properties for tracking intellectual property information. It distinguishes between authors, curators, and contributors. PAV also distinguishes between retrieving a resource 'as is', importing a resource through a data transformation and accessing a resource. The latter is useful when resources such as webpages are accessed but not cached or imported into the system. PAV also allows us to specify the agent that performed the task and the time when the task was performed. Last, versioning properties are provided to link a version of the resource with the previous one of

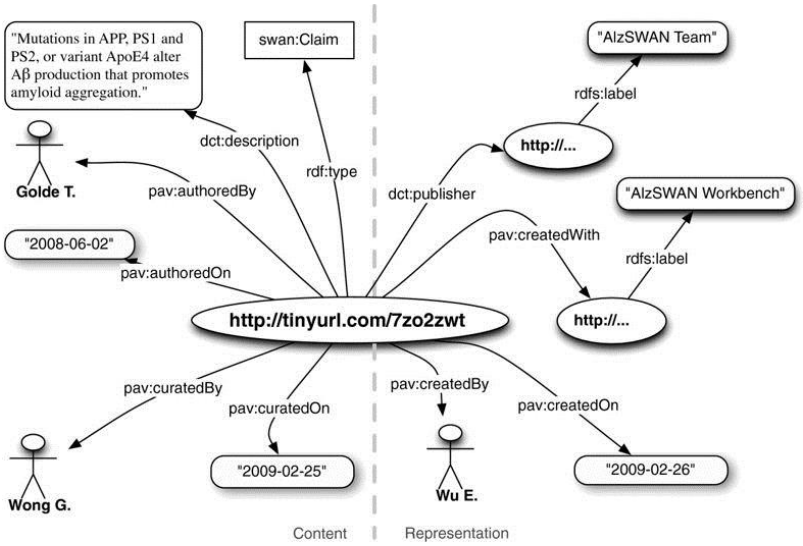

Figure 6: Example illustrating authoring with PAV

the same lineage and indicate an artifact as a derivation of another, not necessarily of the same lineage.

Dublin Core Metadata Initiative (DCMI)

The Dublin Core metadata standard[2] is a simple yet effective set of 15 elements (Figure 7) for describing a wide range of networked resources on the Internet. Although the Dublin Core favours document-like objects, it can be applied to other resources as well. Its suitability for use with particular non-document resources depends to some extent on how closely their metadata resemble typical document metadata and also what purpose the metadata is intended to serve. The following graph re-expresses unqualified DC in HTML in RDF in a schematic way.

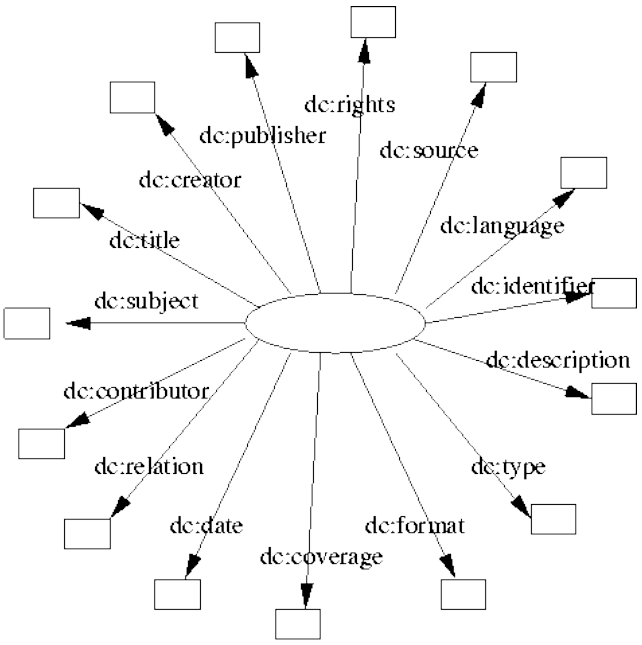

Figure 7: Core DCMI metadata

[2] http://dublincore.org/documents/dces/

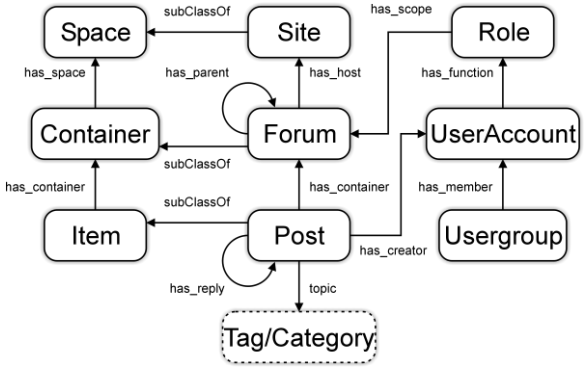

Figure 8: Example illustrating authoring with SIOC

Semantically-Interlinked Online Communities (SIOC)

The SIOC ontology (Passant et al. 2010) has become a popular metadata standard providing the main concepts and properties required to describe information from online communities (e.g., message boards, wikis, weblogs, etc.) on the Semantic Web (Breslin et al. 2005). It also provides methods for interconnecting discussion methods such as blogs, forums and mailing lists to each other (Figure 8).

Simple Knowledge Scheme (SKOS)

The SKOS Core Vocabulary (Miles 2006) is a model for expressing the basic structure and content of concept schemes. The term "concept scheme" is used to describe "a set of concepts, optionally including semantic relationships between those concepts". These concept schemes might include thesauri, classification schemes, subject heading lists, taxonomies, terminologies, glossaries and other types of controlled vocabulary.

For example, most items of a thesaurus could be mapped to a series of `skos:Concepts` containing preferred labels (`skos:prefLabel`) and non-preferred labels (`skos:altLabel`). It may also contain various broader terms (`skos:broader`) or related terms (`skos:related`), and so forth (Figure 9).

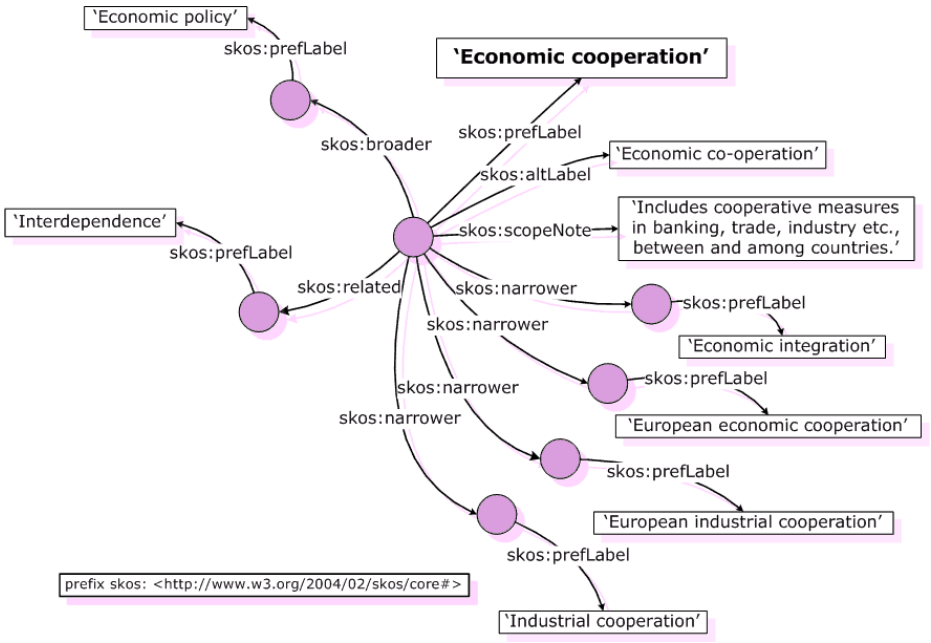

Figure 9: SKOS example

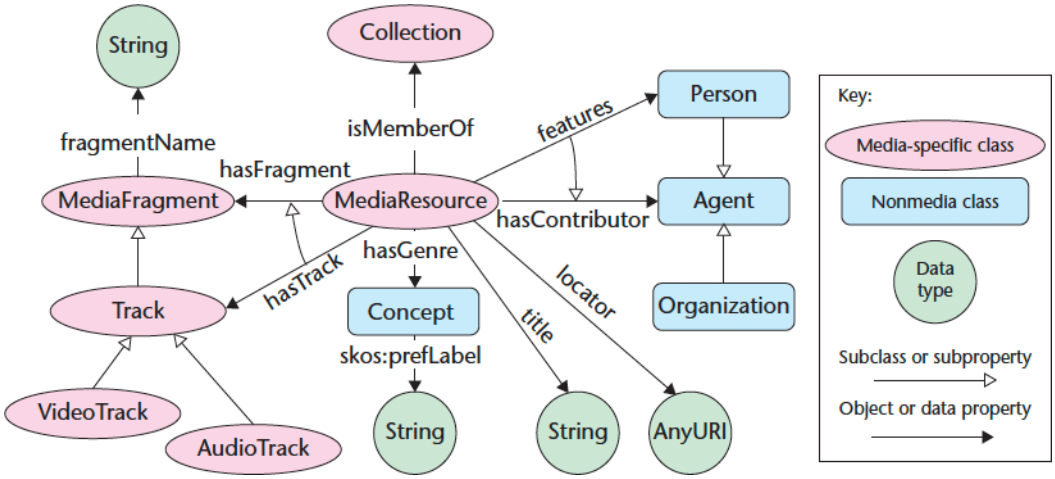

Figure 10: Ontology for Media Resources core model

5.2.3 **Multimedia content**

Ontology for Media Resources

The Ontology for Media Resources 1.0[3] was developed by the W3C Media Annotations Working Group (MAWG) with the purpose to identify a minimum set of core properties necessary and sufficient to describe and retrieve information about media resources (video, audio, images) on the Web. The ontology consists of 20 descriptive properties (*Identification*: identifier, title, language and locator; *Creation:* contributor(s), creator, date and location; *Content description*: description, keyword, genre and ratings; *Relational*: relation and collection; *Rights*: copyright and policy; *Distribution:* publisher and target audience; *Fragment*: fragment and namedFragment) and eight technical properties (such as frame size, duration, compression, format, etc.). The descriptive properties are media agnostic and apply to descriptions of multimedia works (such as movies) that are not specific to an instantiation (an AVI file, for example), while the technical properties are used when describing a particular instantiation of the content. An overview of the core concepts of the Ontology for Media Resources is depicted in Figure 10 (Stegmaier et al. 2013).

Common Shape Ontology

The Common Shape Ontology conceptualizes knowledge about "digital shape" resources, ranging from 2D/3D images to videos, 3D models and 3D Animations (Vasilakis et al. 2007). The ontology lies at an intermediate level between top ontologies and domain ontologies being specific and detailed enough to be used and instantiated directly, but also general enough to constitute the foundation for domain- specific ontologies. The most important concept is the `ShapeRepresentation` class, whose instances are the actual digital shapes. The ontology consists of properties describing the creator, the owner, the contact and the uploader of a digital shape. Since the granularity of these roles is often not well defined, the range of the above relations is `PersonInfo` and `InstitutionInfo`, which in turn can be mutually linked by the relation `worksFor`. Another way to look at a digital shape is to consider it as a file. For this reason each shape can be related to a `FileInfo` instance, in

[3] http://www.w3.org/TR/mediaont-10

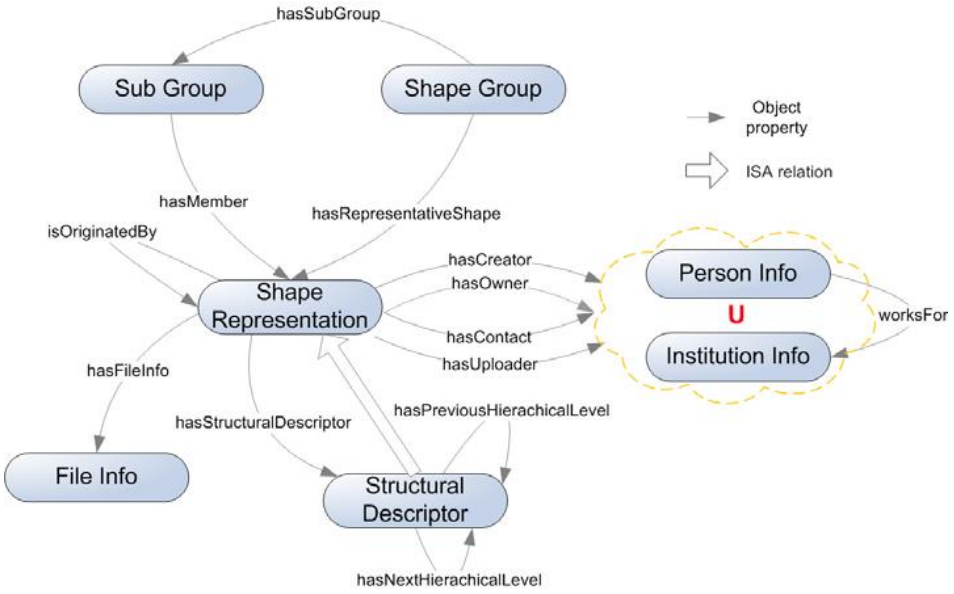

Figure 11: Common Shape Ontology structure

which the information about the name, the size, the format and the URL of the file are stored. An overview of the Common Shape Ontology structure, where only the most important concepts are shown, is given in Figure 11.

3D Modelling Ontology

Both the MPEG-7 and the X3D standard[4] define their terms in a semi-structured XML Schema-based (XSD) vocabulary, which is machine-processable, but not machine-interpretable (L. F. Sikos 2017). This limitation can be addressed by using Semantic Web standards, such as RDF, RDF Schema and OWL (L.F. Sikos 2015), so that the structured representation of the corresponding concepts becomes machine-interpretable by linking them to their formal definitions and other, related concepts from the Linked Open Data (LOD) Cloud. For this reason, several attempts have been made to map the XML Schema of MPEG-7 to RDFS and OWL (Leslie F Sikos and Powers 2015) and the XML Schema of X3D to OWL (OntologyX3D (Kalogerakis, Christodoulakis, and Moumoutzis 2006), and more recently, the 3D Modelling Ontology (3DMO)[5], which not only maps the X3D vocabulary to OWL, but also extends it with important terms from the 3D modelling industry.

To date, the 3D Modelling Ontology is the most expressive 3D ontology, which is defined in the SROIQ(D) description logic. It utilizes the DL syntax and semantics to define 3D modelling concepts, such as geometry, material, texture, environment, 3D objects, and polygons, in a taxonomic structure, together with roles (in a hierarchy), individuals and relations defined with Schema.org, Dublin Core, and FOAF concepts.

5.3 Annotation models

Annotating, the act of creating associations between distinct pieces of information, is a pervasive activity online in many guises. Annotations are typically used to convey information about a resource or associations between resources. Simple examples include a

[4] http://www.web3d.org/x3d/what-x3d/
[5] http://3dontology.org

comment or tag on a single web page or image, or a blog post about a news article. In this section, we present two annotation models, the Web Annotation Data Model and the Europeana Data Model which have inspired the V4Design annotation model described in Section 6.1.

5.3.1 Web Annotation Data Model

The Web Annotation Data Model[6] specification describes a structured model and format to enable annotations to be shared and reused across different hardware and software platforms. This interoperability may be either for sharing with others, or the migration of private annotations between devices or platforms. The shared annotations must be able to be integrated into existing collections and reused without loss of significant information. Common use cases can be modelled in a manner that is simple and convenient, while at the same time enabling more complex requirements, including linking arbitrary content to a particular data point or to segments of timed multimedia resources.

The specification provides a specific JSON format for ease of creation and consumption of annotations based on the conceptual model that accommodates these use cases, and the vocabulary of terms that represents it. The Web Annotation Vocabulary[7] specifies the set of RDF classes, predicates and named entities that are used by the Web Annotation Data Model. It also lists recommended terms from other ontologies that are used in the model, and provides the JSON-LD Context and profile definitions needed to use the Web Annotation JSON serialization in a Linked Data context.

An annotation is considered to be a set of connected resources, typically including a body and target, and conveys that the body is related to the target. The exact nature of this relationship changes according to the intention of the annotation, but the body is most frequently somehow "about" the target. This perspective results in a basic model with three parts, depicted in Figure 12. The full model supports additional functionality, enabling content to be embedded within the annotation, selecting arbitrary segments of resources, choosing the appropriate representation of a resource and providing styling hints to help clients render the annotation appropriately. Annotations created by or intended for machines are also possible, ensuring that the Data Web is not ignored in favour of only considering the human-oriented Document Web.

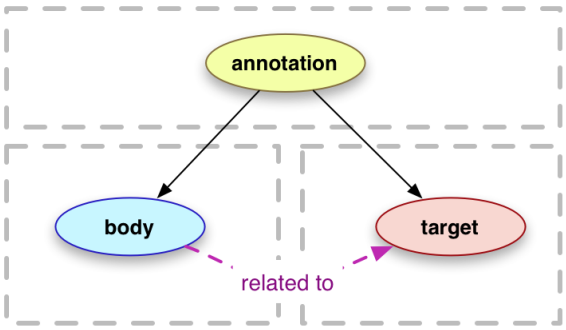

Figure 12: Core Web Annotation Data Model pattern

[6] https://www.w3.org/TR/annotation-model/
[7] https://www.w3.org/TR/annotation-vocab/

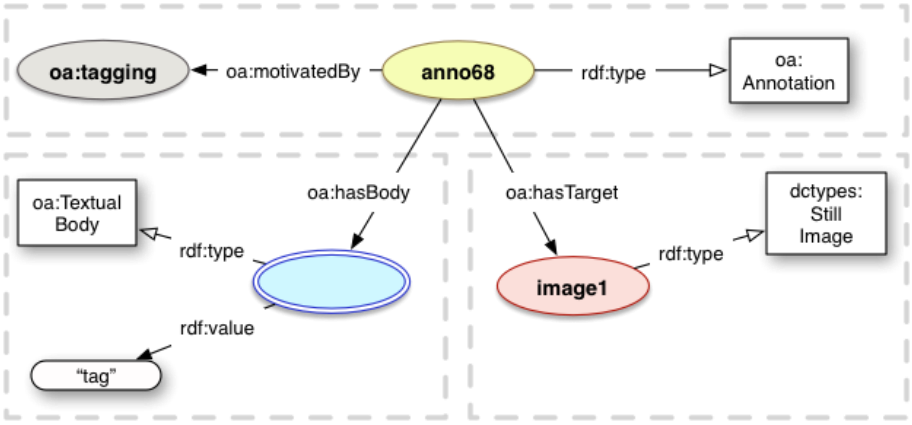

Figure 13: Example annotation of an image

The Web Annotation Data Model does not prescribe a transport protocol for creating, managing and retrieving annotations. Instead it describes a resource oriented structure and serialization of that structure that could be carried over many different protocols. Figure 13 depicts an example annotation of an image, using the Web Annotation Data Model.

5.3.2 Europeana Data model

The Europeana Data Model (EDM) (Doerr et al. 2010) has been proposed for structuring the data that Europeana ingests, manages and publishes. The model is not built on any particular standard but rather adopts an open and scalable approach that can accommodate the range and level of details of particular standards such as LIDO for museums, EAD for archives or METS for digital libraries. Moreover, the model not only supports the level of detail of the content providers' metadata but also enables data enrichment from a range of third party sources.

The rationale behind EDM is that it makes a distinction between the object this structure is about and the digital representation of the object, which can be accessed over the Web. It also adheres to the modelling principles of the Semantic Web enabling various fine-grained models to be attached. This assumption is conceptually in line with the specification of the Web Annotation Data Model described in the previous section.

In more detail, EDM provides three core classes to enable the representation and accessing of objects provided to Europeana, together with their digital representations, which is regarded as one logical whole. In addition, EDM introduces and re-uses metadata properties to semantically enrich objects and connecting them to other resources while it allows for different levels of granularity in the descriptions. It also provides support for ingesting the descriptive metadata submitted by various providers for the same object and representing new information added by Europeana.

Representing provided data as aggregations

EDM has three core classes of resources:

- **edm:ProvidedCHO:** captures the "provided cultural heritage object" which can be a painting, a movie, a music score, a book, etc. and is mapped to a `edm:ProvidedCHO` concept acting as an identifier for the "real" object.
- **edm:WebResource:** refers to the, one or more, accessible digital representations of this object, some of which will be used as previews, such as the digital picture of a

painting. This digital representations of the object is mapped to a `edm:WebResource` concept.

- **ore:Aggregation:** denotes the aggregation of the provided object, together with its digital representations, which represents the result of this provider's activity

The properties for interconnecting these classes are the following:

- **edm:aggegatedCHO**, which relates `ore:Aggregation` with one resource that stands for the provided object using the `edm:aggegatedCHO` property and
- **edm:hasView**, which relates `ore:Aggregation` with one or more resources that are digital representations of the provided object, using the `edm:hasView` property.

Both `edm:aggregatedCHO` and `edm:hasView` properties are sub-properties of `ore:aggregates`, representing the fact that the aggregation indeed aggregates the "real" object and its digital views. Figure 14 depicts a high-level view of the core classes and properties linking them.

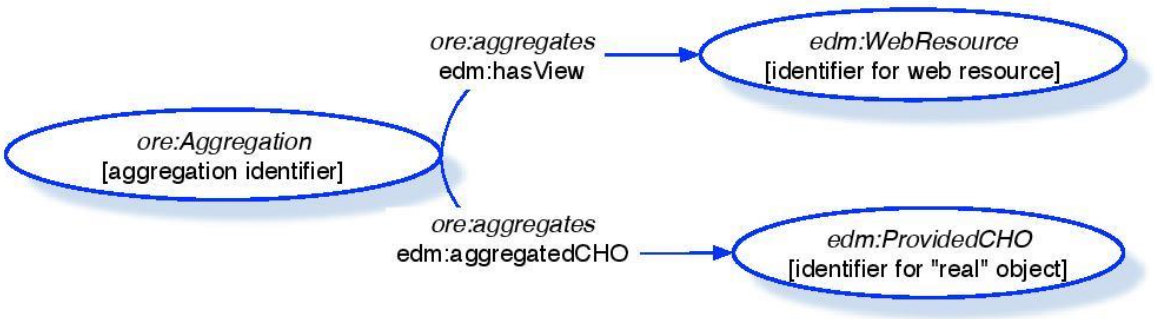

Figure 14: Visualization of the three core EDM classes for data providers

EDM enables capturing a description of the "digital environment" of an object submitted to Europeana, and attaching descriptive information to the various resources that take part in this environment. To this end, EDM includes a set of "descriptive" and "contextual" properties that capture the different features of a resource, as well as relate it to the other entities in its context. These properties can be either introduced as new EDM-related properties or re-used by existing vocabularies, such as the `dc` and `dcterms` properties of DCMI, which are used to directly link text values to the object.

5.4 Discussion

This section has introduced the basic notions underlying the Semantic Web and provided a brief overview of key technologies empowering the envisaged knowledge sharing and reuse across heterogeneous environments. Expressive ontology languages allow the elegant capture of complex knowledge and its semantics in a formal way, rendering it amenable to automated reasoning tasks with well-understood computational properties. Rules augment further the expressive capabilities, by allowing the representation of richer semantic relationships. In addition, we briefly presented existing ontologies relevant to the V4Design application domain, focusing on the provided ontology constructs and design patterns. As such, ontologies for capturing events and observations have been reviewed. Such ontologies can be used to model analysis results from other modules as observations. In addition, general purpose ontologies, such as PROV-O, and metadata vocabularies, such as SKOS and

DCMI, provide useful annotation schemata. In addition, a number of ontologies exist for annotating multimedia content, such as images, video and 3D models. Finally, two annotation models have been reviewed that allow the formal annotation of entities with metadata.

The ontologies reviewed in this section served us as valuable references for distilling the advantages and disadvantages of alternative modelling solutions and the trade-offs and restrictions pertinent to different scopes, before making our modelling choices. For example, for the modelling and formalisation of descriptive information, our choices have been largely shaped by the DCMI and schema.org vocabularies. In addition, the V4Design annotation model capitalises on the Web Annotation Data model, as we describe in Section 6.1.

The formalisation has been performed keeping also in mind the need to have an ontology that will support the planned reasoning tasks in T5.2. Since many of the exact reasoning-incurring dependencies could not be specified during this phase of ontology building, we opted for a concise modelling that covers the foundational notions identified through the competency questions, while in parallel enforcing modularity and separation of concerns so that extensibility and future ontology updates are facilitated.

6 V4DESIGN ANNOTATION MODEL AND REASONING FRAMEWORK

6.1 V4Design Annotation Model

In line with the preceding requirement analysis of V4Design application contexts, a number of ontological constructs have been defined in order to support data modelling, integration and reasoning over the distilled information. These include:

- Constructs for capturing metadata of different resources, such as aesthetics, recognised buildings and objects, named entities and concepts, etc.
- A structured model and format to enable annotations and assertions to be defined, shared and reused across both inside the V4Design application context but also in different hardware and software platforms.

A key design choice underpinning the engineering of the V4Design models has been the adherence to a pattern-based approach, so as to capitalise on a modular, extensible and interoperable framework for expressing annotations and achieve a better degree of knowledge sharing, reuse and interoperability. In particular, the V4Design annotation pattern reuses the Web Annotation Data Model whose a brief summary is presented in Section 5.3.1. It also reuses a number of existing schemata, such as DCMI and schema.org to inherit general purpose hierarchies and descriptive attributes (see Appendix A.2).

It must be noted that as the modelling and reasoning requirements evolve, as well as the user requirements and output of component become richer, iterative cycles of assessment and respective revisions will take place. These will mainly affect the domain models that we use to capture the various information types generated within V4Design. The pattern-based approach ensures that the conceptual model for associating resources with annotations will not be affected by the updated domain models. This is especially important since it allows incremental and targeted updates to be performed on the underlying vocabularies (according to the updated requirements), minimising the risk for compatibility errors and the impact that these changes may have on the platform.

In this section, we present the way the Web Annotation Data Model is used to address the V4Design modelling requirements, associating the media types that are generated by the V4Design modules with metadata, we call views (Figure 15). It should be mentioned that the annotation model and underlying ontologies are checked against the requirements in order to ensure that they adequately cover the knowledge that they are expected to capture. As a consequence, formalisation and revision activities have been carried on iteratively, and will continue for the remaining duration of the project, as the use cases and requirements evolve. As already mentioned, the separation of the domain ontologies from the pattern used for attaching metadata to various resources in the form of views fosters reusability, extensibility and interoperability, minimising the

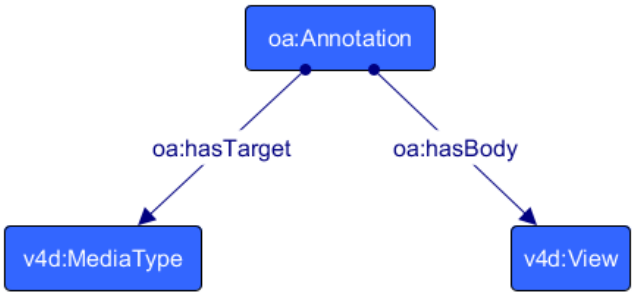

Figure 15: Core annotation model in V4Design

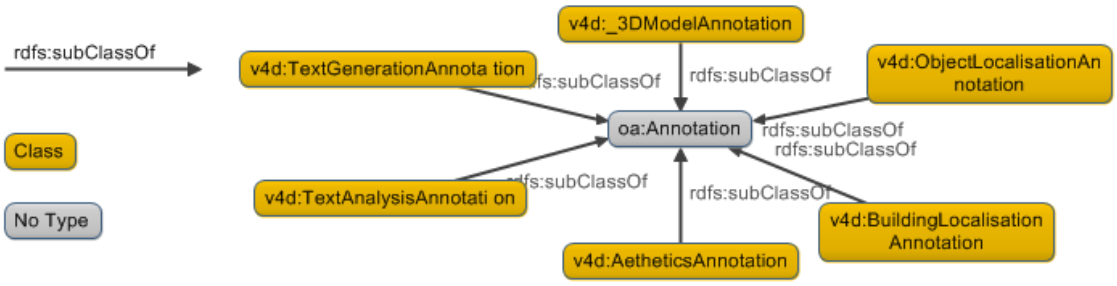

Figure 16: The six annotation classes in V4Design

effort needed to incorporate updates needed due to updated user and technical requirements.

6.1.1 V4Design Annotation Classes

V4Design extends the `oa:Annotation` class, defining 6 domain specific annotations classes that are depicted in Figure 16. Each of these classes is used as the root annotation resource, attaching metadata views to media types.

There are four main media types in V4Design, for which WP5 gets analysis results from other

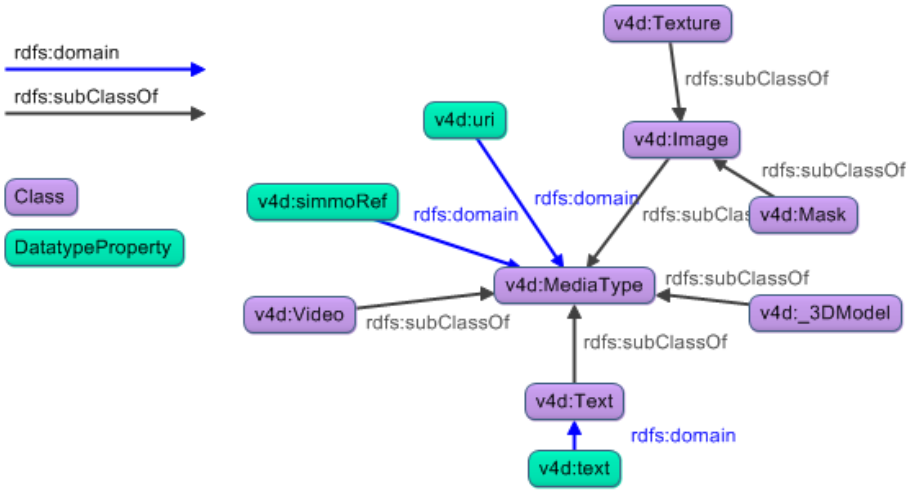

Figure 17: Ontology for media types

components of the architecture: videos, images (masks, textures), text and 3D models. Figure 17 depicts the basic hierarchy of media types. The ontology also contains descriptive properties that capture basic metadata that characterise assets and media types, i.e. attributes that are not subject to different interpretations and do not derive from analysis, but they are static. Ids, licence information and timestamps are some examples of such metadata that the V4Design ontology needs to support. There are already existing ontologies that provide a common vocabulary for such attributes, such as the Dublin Core Metadata or the shema.org vocabulary, that V4Design ontology reuses. Figure 17 depicts an excerpt of the media type ontology.

Each annotation class restricts the values of the `oa:hasTarget` and `oa:hasBody` properties. In other words, it associates the target of the annotation (i.e. the media type), with the metadata view, i.e. the RDF graph that contains the metadata that have been derived by the

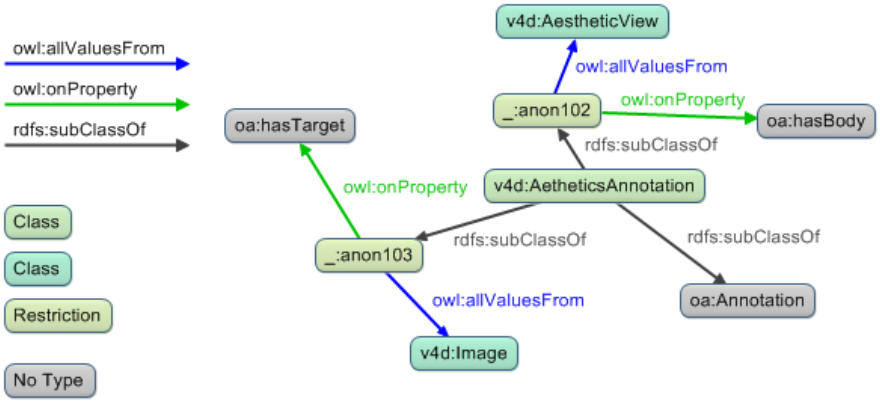

Figure 18: Annotation class for aesthetics

analysis. In the following, we present the basic structure of the respective annotation models.

Aesthetics Annotation Class

The initiation of an annotation relevant to aesthetics is performed by defining instances of the `v4d:AestheticsAnnotation` class. As depicted in Figure 18, this class restricts the `oa:hasBody` property to take as values only instances of the `v4:AestheticView` class. In addition, the `oa:hasBody` property is restricted to take as values only instances of the `v4d:Image` media type, since aesthetics are derived only from images.

Object Localisation Annotation Class

The initiation of an annotation relevant to object localisation results is performed by defining instances of the `v4d:ObjectLocalisationAnnotation` class. As depicted in Figure 19, this class restricts the `oa:hasBody` property to take as values only instances of the `v4:LocalisationObjectView` class. In addition, the `oa:hasBody` property is restricted to take as values only instances of the `v4d:Image` and `v4d:Video` media types, since object localisation is performed over both media types.

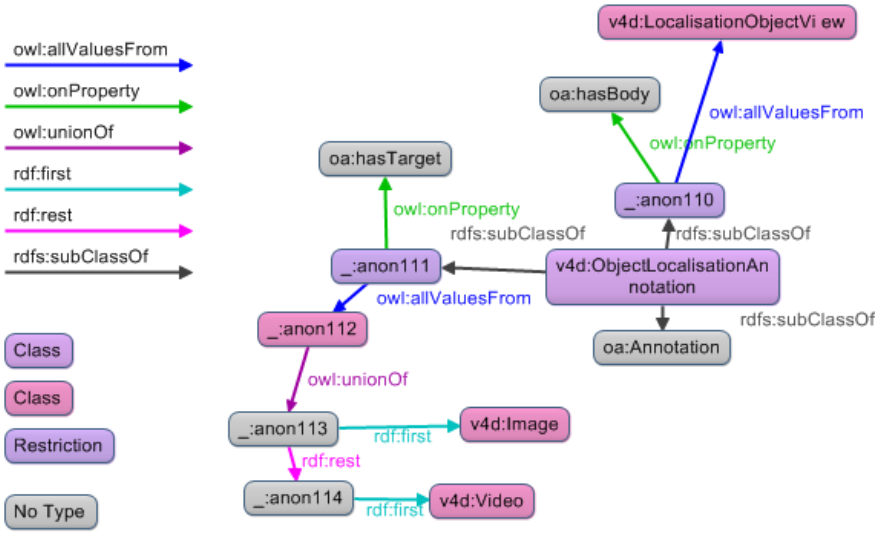

Figure 19: Annotation class for object localisation

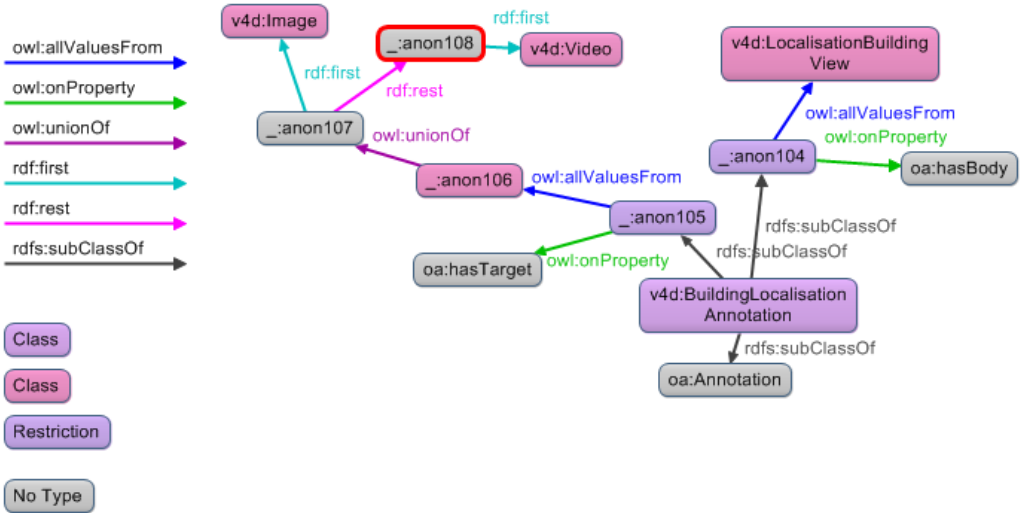

Figure 20: Annotation class for building localisation

Building Localisation Annotation Class

The initiation of an annotation relevant to building localisation is performed by defining instances of the `v4d:BuildingLocalisationAnnotation` class. As depicted in Figure 20, this class restricts the `oa:hasBody` property to take as values only instances of the `v4:LocalisationBuildingView` class. In addition, the `oa:hasBody` property is restricted to take as values only instances of the `v4d:Image` and `v4d:Video` media types, since object localisation is performed over both media types.

Text Analysis Annotation Class

The initiation of an annotation relevant to textual analysis is performed by defining instances of the `v4d:TextAnalysisAnnotation` class. As depicted in Figure 21, this class restricts the `oa:hasBody` property to take as values only instances of the `v4:TextAnalysisView` class. In addition, the `oa:hasBody` property is restricted to take as values only instances of the `v4d:Text` media type.

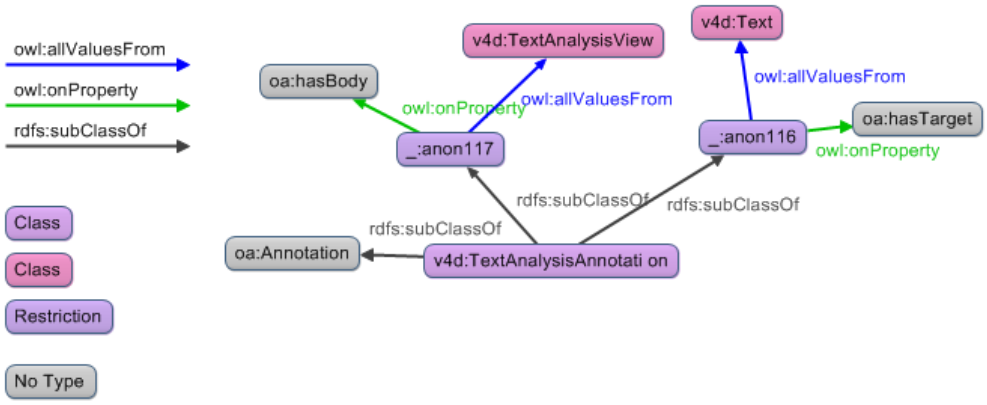

Figure 21: Annotation class for text analysis

Text Generation Annotation Class

The initiation of an annotation relevant to textual analysis is performed by defining instances of the `v4d:TextGenerationAnnotation` class. As depicted in Figure 22, this class restricts the `oa:hasBody` property to take as values only instances of the `v4:TextGenerationView` class.

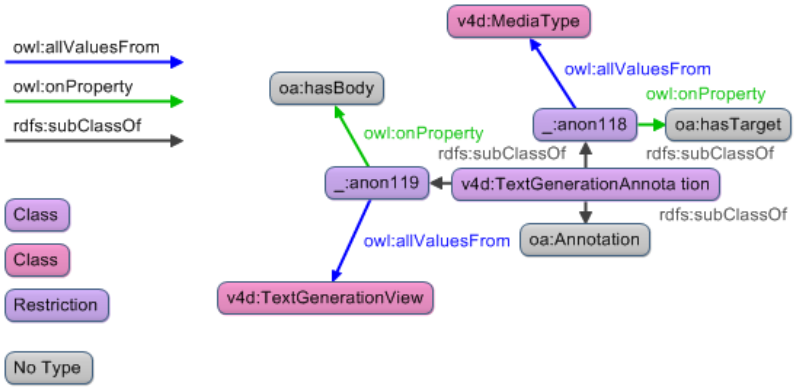

Figure 22: Annotation class for text generation

In addition, the `oa:hasBody` property is restricted to take as values only instances of the `v4d:MediaType` class.

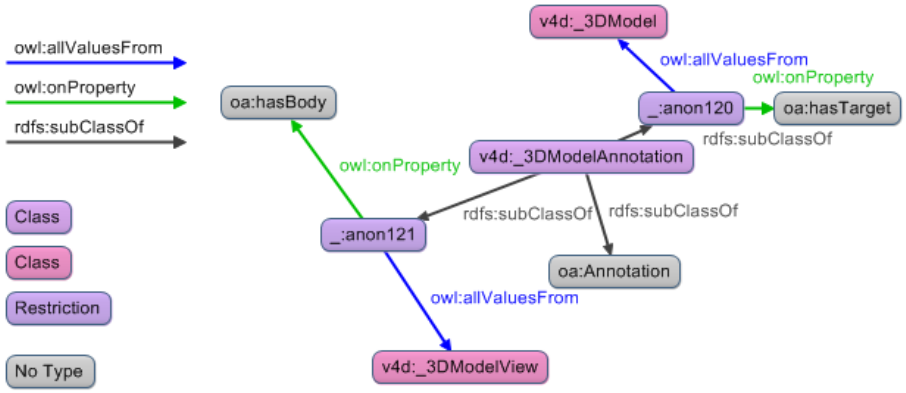

Figure 23: Annotation class for 3D model reconstruction

3D Model Reconstruction Annotation Class

The initiation of an annotation relevant to the reconstruction of a 3D model is performed by defining instances of the `v4d:3DModelAnnotation` class. As depicted in Figure 23, this class restricts the `oa:hasBody` property to take as values only instances of the `v4:3DModelView` class. In addition, the `oa:hasBody` property is restricted to take as values only instances of the `v4d:3DModel` class.

6.1.2 V4Design Views

In V4Design, views are container classes for annotations that are used in `oa:hasBody` property assertions. They are the constructs that capture the actual metadata that the various V4Design modules generate. An expert of the View graph for all annotation classes presented so far can be found in Figure 24. It should be noted that in the current version of the V4Design annotation model, the views contain a very limited number of annotation properties, in line with the modelling requirements of the operational prototype (MS2/M12). As we illustrate through a detailed example in Section 7, the current version of the V4Desing ontological modules is able to effectively capture all the annotation metadata that the V4Design modules generate. As the analysis modules become more advanced and

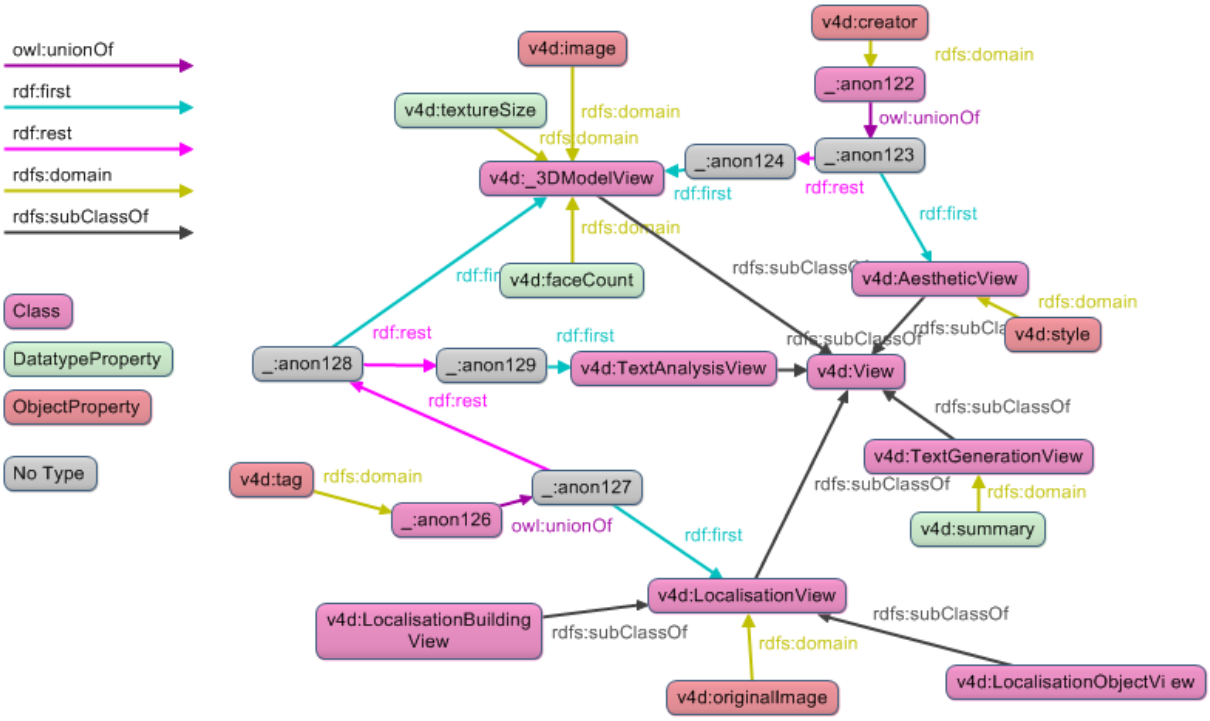

Figure 24: Excerpt of the V4Design View graph

provide richer metadata, the V4Design vocabulary will be updated to fully support the construction of V4Design knowledge graphs.

For each annotation class described in the previous sections, there is a corresponding view class.

Aesthetics

V4Design aims at the extraction of aesthetics, i.e. the categorisation of the aesthetics of paintings and images that contain architecture objects and buildings based on their style (i.e. impressionism, cubism and expressionism), creator and emotion that they evoke to the viewer and combine them so as to produce/suggest novel textures. Based on the feedback we obtained from WP3, Table 3 depicts the styles and creators that are currently supported.

Styles	Baroque, Impressionism, Expressionism, Cubism, Rococo, Minimalism, Abstract Expressionism, Action painting, Analytical Cubism, Art Nouveau, Colour Field Painting, Contemporary Realism, Early Renaissance, Fauvism, High Renaissance, Mannerism Late Renaissance, Naive Art Primitivism, New Realism, Northern Renaissance, Pointillism, Pop Art, Post Impressionism, Realism, Romanticism, Symbolism, Synthetic Cubism, Ukiyo-e
Creators	Salvador Dali, Vincent Van Gogh, Pablo Picasso, Albrecht Durer, Boris Kustodiev, Camille Pissarro, Childe Hassam, Claude Monet, Edgar Degas, Eugene Boudin, Gustave Dore, Ilya Repin, Ivan Aivazovsky, Ivan Shishkin, John Singer Sargent, Marc Chagall, Martiros Saryan, Nicholas Roerich, Pierre Auguste Renoir, Pyotr Konchalovsky, Raphael Kirchner, Rembrandt, Paul Cezanne

Table 3: Styles and creators supported by aesthetics extraction

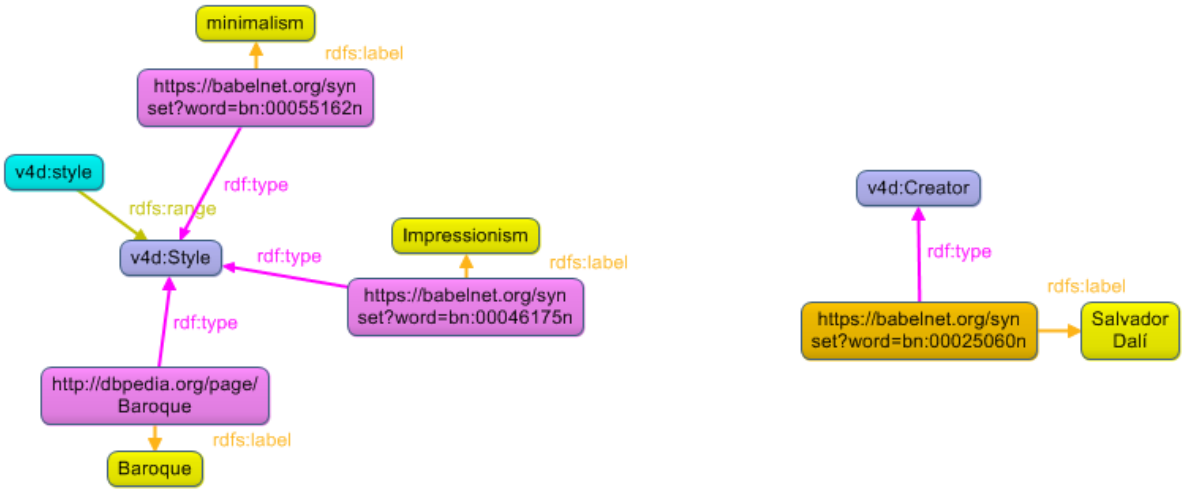

Figure 25: Style and Creator classes with example instances

Two properties have been defined for the creators (`v4d:creator`) and styles (`v4d:style`) whose domain is the `v4d:AestheticView`. In addition, each one style (`v4d:Style`) and creator (`v4d:Creator`) in Table 3 has been mapped to corresponding lexical resource in the BabelNet and/or DBpedia semantic network. For example, "Impressionism" is represented with the resource https://babelnet.org/synset?word=bn:00046175n, while "Salvador Dalí" with the resource https://babelnet.org/synset?-word=bn:00025060n.

Building and Object Localisation

Spatio-temporal building and object localisation in images and video frames aims to define their type, i.e. whether the image or video contains a building, object or a painting and then semantically segment it in a spatio-temporally manner in order to localise the spatial elements of the buildings (i.e. type of window, door, roof, decoration, facade, etc.) and the surrounding area. Based on the feedback we obtained from WP4, Table 4 depicts the building and objects that are currently supported.

Interior objects	bottle, plate, wine glass, cup, fork, knife, spoon, bowl, chair, couch, potted plant, bed, mirror, dining table, window, desk, toilet, door, sink, vase
Exterior objects	person, bicycle, car, motorcycle, airplane, bus, train, truck, boat, traffic light, fire hydrant, street sign, stop sign, parking meter, bench, bird, cat, dog, horse, sheep, cow, elephant, bear, zebra, giraffe, hat, backpack, umbrella, shoe, eye glasses, handbag, tie, suitcase, frisbee, skis, snowboard, sports ball, kite, baseball bat, baseball glove, skateboard, surfboard, tennis racket
Buildings	abbey, alley, amphitheatre, apartment building - outdoor, aqueduct, arcade, arch, atrium, auditorium, balcony - exterior, balcony - interior, barn, barn door, basilica, beach house, bistro - outdoor, boathouse, bookstore, booth-indoor, bow window - indoor, bridge, building facade, bullring, burial chamber, bus station-indoor, butchers shop, cabin-outdoor, cafeteria, campus, candy store, castle, catacomb, cathedral-indoor, cathedral-outdoor, cemetery, chalet, chapel, church-indoor, church-outdoor, clothing store, coffee shop, corral, corridor, cottage, courthouse, courtyard, dam, department store, diner-outdoor, dock, doorway-outdoor, downtown, embassy, entrance hall,

	excavation, fast food restaurant, fire station, fountain, garage-indoor, garage-outdoor, gas station, gazebo-exterior, general store-outdoor, gift shop, greenhouse-outdoor, hallway, harbour, hospital, hotel-outdoor, house, hunting lodge-outdoor, igloo, industrial area, industrial park, inn-outdoor, jewellery shop, kasbah, library-indoor, library-outdoor, lift bridge, lighthouse, loading dock, lobby, mansion, manufactures home, market-outdoor, mausoleum, medina, mezzanine, moat-water, monastery-outdoor, mosque-outdoor, motel, museum-indoor, museum-outdoor, natural history museum, oast house, office building, pagoda, palace, parking garage-outdoor, pavilion, pharmacy, pier, playground, plaza, porch, promenade, pub-indoor, pulpit, racecourse, residential neighbourhood, restaurant, ruin, schoolhouse, shed, shop front, shopping mall-indoor, ski lodge, ski resort, skyscraper, stable, stadium-baseball, stadium-football, stadium-soccer, staircase, swimming pool-indoor, swimming pool-outdoor, synagogue-outdoor, temple-east Asia, temple-south Asia, throne room, tower, tree house, trench, veranda, viaduct, village, water tower, wind farm, windmill, yard

Table 4: Objects and buildings recognised by building and object localisation.

Each one object type and building in Table 4 has been mapped to corresponding lexical resource in the BabelNet and DBpedia semantic network. For example, "bottle" is represented with the resource https://babelnet.org/synset?word=bn:00012339n, while "abbey" with the resource https://babelnet.org/synset?word=bn:00000234n (Figure 26). It should be noted that some values have been mapped to BabelNet concepts and not the complete list, since the list of objects and buildings is under development. As depicted in Figure 24, the `v4d:LocalisasationView` uses the property `v4d:tag` to associate recognised images and building with the video and image media types.

Text Analysis

Text analysis addresses the analysis and capture of the natural language textual material into structured representations, so that appropriate system responses can subsequently be inferred and textual summaries can be produced. For example, text analysis aims to semantically analyse the captions and/or descriptions of assets (video titles, paining descriptions, image captions) in order to extract named entities, concepts and relations that can be used to enrich the semantic signature of the asset.

The output of text analysis is already semantically annotated, i.e. the detected concepts are

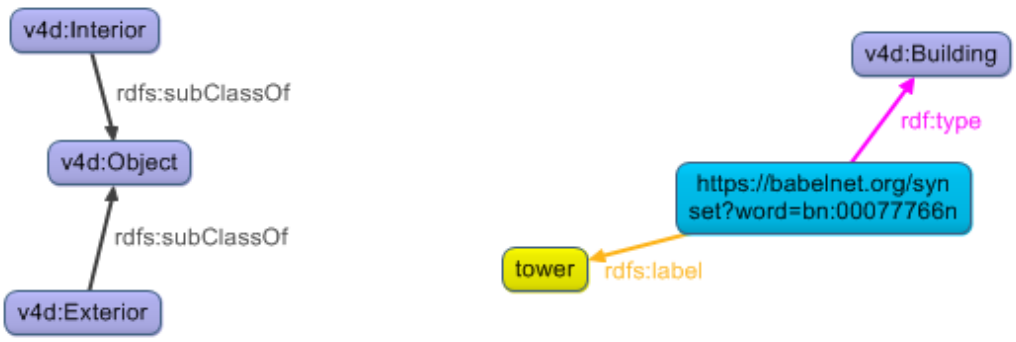

Figure 26: Object hierarchy and Building example

associated with DBpedia and/or BabelNet resources. The V4Design core vocabulary needs only to capture these annotations and associate them with the original text and possible to the media type (video or image) that tis textual content is related with through instances of the `v4d:TextAnalysisView`. This pattern for associating semantics with assets and media types is described in Section 7 and it is common for all types of resources.

Text Generation

Text generation generates textual reports, descriptions, or summaries starting from annotations extracted from text, webpages, and/or visual analysis. It starts from abstract representations, modelled, e.g., as RDF triples, which are stored in the Knowledge Base. Similarly to text analysis, the output of text generation does not need a particular vocabulary. The only requirements is to associate the generated texts with the assets and media types (through the `v4d:TextGenerationView`), which is described in Section 7.

3D Models

3D model reconstruction is responsible for conversion of input video and image data into 3D point clouds and meshes. Apart from the actual object, this task also generates a number of metadata, such as the number of point clouds, the initial source of reconstruction (video or the set of images), as well as features, such as quality.

The `v4d:3DModelView` acts as a container for capturing metadata generated by the 3D model reconstruction module. Currently, only three such metadata properties are defined, namely `v4d:faceCount`, `v4d:textureCount` and `v4d:image`. The first two are datatype properties, while the last one is an object property that associates the 3D model with the images that have been used to generate it. An example is presented in Section 7 about the way these properties can be instantiated in a 3D model view.

6.1.3 KB Population

As depicted in Figure 1, WP5 encapsulates the KB population module, which is responsible for translating incoming data into RDF-based annotations, following the annotation model we described in the previous sections. More specifically, for each input type (aesthetics, building and object localisation, text analysis, text generation and 3D model reconstruction), WP5 implements a mapping services pertinent to the format and structure of the data that is provided as input (Figure 27). The logic behind the translation is straightforward and examples are given in Section 7.

6.1.4 Other annotation properties

As we have already described, the technical requirements (D6.2) are still under development, therefore the exact capabilities and outputs of the V4Design components have not been finalised yet. The annotation properties described so far have been mainly elicited through the simulation example we present in Section 7 and aim at capturing the analysis results of the current development cycle towards MS2. In Table 5, we present a list of pending annotation properties that are not yet part of the V4Design Views, but there will be included in the model in next development and validation cycles, towards the first prototype.

Annotation property	Mappings
v4d:licence	http://purl.org/dc/terms/license http://schema.org/license
v4d:date	http://purl.org/dc/elements/1.1/date http://purl.org/dc/terms/created
v4d:lng	http://schema.org/longitude
v4d:lat	http://schema.org/latitude
v4d:location	http://schema.org/location
v4d:language	http://purl.org/dc/terms/language
v4d:material	http://schema.org/material
v4d:scale	http://purl.org/ontology/x3d/scale
v4d:size	http://purl.org/ontology/x3d/size
v4d:width	http://schema.org/width
v4d:height	http://schema.org/height
v4d:format	http://purl.org/dc/terms/format
v4d:timestamp	http://purl.org/dc/elements/1.1/date
v4d:thumbnail	http://schema.org/thumbnail
v4d:relevantAssets	http://purl.org/dc/terms/relation
v4d:subModel	http://purl.org/dc/terms/isPartOf

Table 5: Pending annotation properties to be included in the next development cycle

6.2 Ontology-based Reasoning Framework

So far, the focus has been mainly on the identification of V4Design's key modelling requirements and the development of pertinent vocabularies to support the representation and mapping of content on semantic knowledge structures.

In this section, we present the preliminary version of WP5's reasoning framework (towards MS2/M12), which aims at the intelligent aggregation of the metadata collected from the various V4Design modules by combining, integrating and semantically interpreting knowledge captured in the KB. The reasoning techniques employed by WP5 at this stage constitute the first preliminary approach to address the requirements, with the rest pending for later prototypes. More elaborate and flexible reasoning and interpretation schemes will be tackled in future versions of the framework, as V4Design components mature, which will allow in turn for more sophisticated interpretations that will be reported in upcoming deliverables.

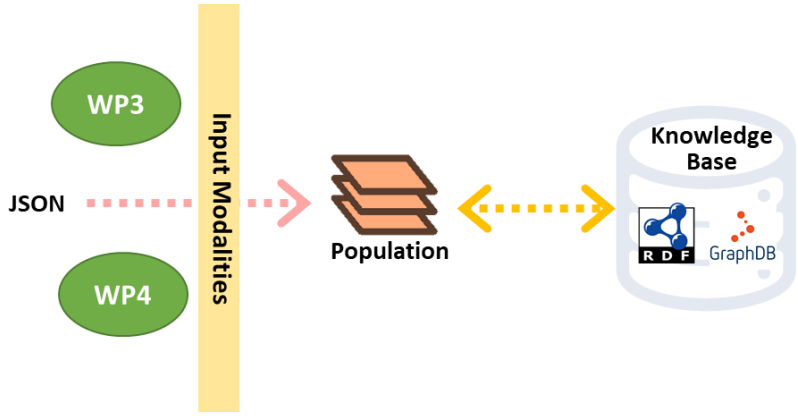

Figure 27: KB population service

6.2.1 **Reasoning Architecture**

The core elements of the reasoning system are depicted in Figure 28. All in all, the framework extends the semantics of the V4Design's conceptual models (i.e. the Annotation Model) with rules that, based on the available context, i.e. the metadata collected from the analysis tasks, further update the KB.

The reasoning framework heavily depends on the semantics of the V4Design Annotation Model described in Section 6.1. The semantics is used to acquire a preliminary understanding of the available content and the dependencies among the multimodal results in the form of knowledge graphs that interlink metadata. These knowledge graphs are then used as input to the reasoning framework that triggers the necessary reasoning procedure (rules) to derive additional relations. As such, the reasoning framework can be viewed as a hybrid data integration and interpretation scheme, where ontologies and rules incrementally couple dynamic information.

Reasoning is performed over the Knowledge Base (see Figure 1) where all the metadata of the V4Design pipeline are stored. For the prototype implementation, the GraphDB triple store[8] has been used to implement the Knowledge Base. It is a highly scalable RDF triple store that provides native OWL 2 RL reasoning services[9] and SPARQL-based query

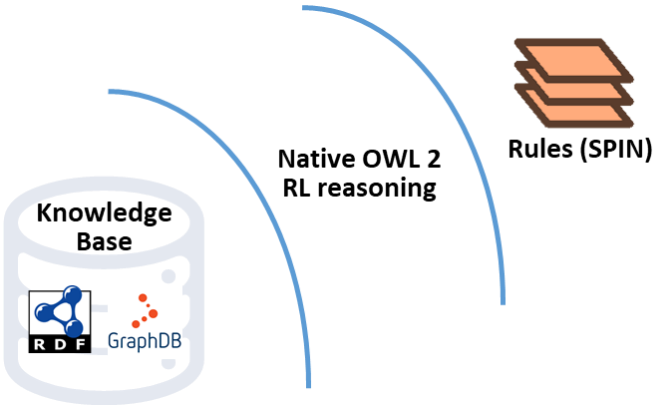

Figure 28: Abstract reasoning architecture

[8] http://graphdb.ontotext.com/
[9] https://www.w3.org/TR/owl2-profiles/#Reasoning_in_OWL_2_RL_and_RDF_Graphs_using_Rules

interfaces[10]. The native OWL 2 RL reasoning ensures that the semantics of the OWL 2 language is fully supported, such as the transitivity of subclass relations. However, the native OWL 2 reasoning services provide limited expressivity and are not able to handle complex domain relations. For example, the semantics of OWL 2 does not allow the modelling of relations among instances that do not follow the tree model property. In addition, the native DL semantics does not allow the dynamic generation of new individuals.

Apart from semantically analysing and correlating metadata, reasoning will also provide advanced searching capabilities to the end users. For example, the parameters of user queries (e.g. filtering by keywords or tags) will be handled by the reasoning framework in order to formulate the necessary queries to retrieve metadata from the KB and send responses back. This advanced query formulation service will be not part of the operational prototype (MS2/M12) and it will be integrated in the first prototype (M3/M18).

6.2.2 Inference Rules

We use SPIN rules, i.e. SPARQL construct graph patterns, to implement expressive reasoning rules, enabling property value propagation and instance generation (when needed). The core idea is to associate each reasoning task with one or more SPARQL rules that address specific reasoning requirements, e.g. to propagate aesthetics from images to the 3D models. In the following, we present examples of such reasoning cases and rules. More elaborate rule-based reasoning cases will be tackled in future versions of the prototype framework and reported in upcoming deliverables.

Enriched 3D models with aesthetics

As described in Section 6.1.2, V4Design extracts and categorises the aesthetics of paintings and images that contain architecture objects and buildings based on their style (i.e. impressionism, cubism and expressionism), creator and emotion that they evoke to the

```
PREFIX oa: <http://www.w3.org/ns/oa#>
PREFIX : <https://v4design.eu/ontologies/>
CONSTRUCT {
    [] a :_3DModelView;
        :tag ?style.
}
WHERE {
    ?annotation1 a :AestheticsAnnotation;
                 oa:hasBody ?view1;
                 oa:hasTarget ?image.
    ?annotation2 a :_3DModelAnnotation;
                 oa:hasBody ?view2.
    ?view2 a :_3DModelView;
           :image ?image.
    ?view1 a :AestheticView;
          :style ?style.
}
```

Figure 29: SPARQL rule for 3D model annotation enrichment with aesthetics

[10] https://www.w3.org/TR/sparql11-overview/

viewer. At the same time, 3D model reconstruction creates 3D models based on a set of images. The inference rule Figure 29 is used to propagate the aesthetics of images, which have been used to reconstruct a 3D model, to the 3D object itself.

More specifically, the rule searches for images that have been annotated with a `?style` and that they have participated in the reconstruction of a 3D model, i.e. both the aesthetic annotation (`?annotation1`) and the 3D model annotation (`?annotation2`) annotate the same target (`?image`). If such a style exists, then it is associated with the annotation of the 3D model, using a `:tag` property assertion.

Indirect annotations

Another way the propagation of analysis results among media types can help is the derivation of indirect annotations, i.e. annotations that do not directly refer to the generated 3D model, but they can be used later in ranking the search results or fetching results to certain user queries that might be relevant to the intended context. For example, the title of a video can provide useful insights about the 3D models that have been reconstructed from that video, even if we cannot assume that all 3D models will be relevant of the textual content of the title. The rule in Figure 30 illustrates the SPARQL rules that combine textual and 3D model annotations.

```
PREFIX oa: <http://www.w3.org/ns/oa#>
PREFIX : <https://v4design.eu/ontologies/>
CONSTRUCT {
    [] a :_3DModelView;
        :tag ?tag.
}
WHERE {
    ?annotation1 a :TextAnnotation;
                 oa:hasBody ?view1;
                 oa:hasTarget ?text.
    ?text :simmoRef ?ref.
    ?annotation2 a :_3DModelAnnotation;
                 oa:hasBody ?view2.
    ?view2 a :_3DModelView;
           :video [:simmoRef ?ref].

    ?view1 a :TextAnalysisView;
          :tag ?tag.
}
```

Figure 30: Propagation of textual analysis results to 3D model annotations

7 ONTOLOGY VALIDATION

We present in this section the instantiation of the V4Design Annotation Model to map the results of V4Design components on a simulation example. More precisely, in order to better understand the modelling requirements in WP5 and to have a preliminary view on the output each component generates, we started with an example image and (manually) went through all the analysis steps of the V4Design pipeline (see D6.2 for more technical details on the pipeline and technical requirements of each component). At each step, the technical partners provided feedback about the generated results (both in terms of the format and content), helping us generate the respective annotation vocabularies. The example image, along with the caption is given below.

Caption: The Eiffel Tower seen from the Champ de Mars

7.1 Building Localisation

The output of building localisation on the image is given below.

```
{
  "simmo": "http://v4design-ds.com/simmo/<ref>",
  "assets": [
    {
      "type": "image",
      "original": "http://v4design-ds.com/file/<imageId>",
      "mask": "https://v4design-ds.com/file/<maskId>",
      "tags": [
        "tower"
      ]
    }
  ]
}
```

More specifically, the module generates a mask for the building (Eiffel Tower) depicted in the picture and associated this mask with a tag (‘tower’). The output also contains descriptive properties relevant to the references and Ids of the underlying data storage.

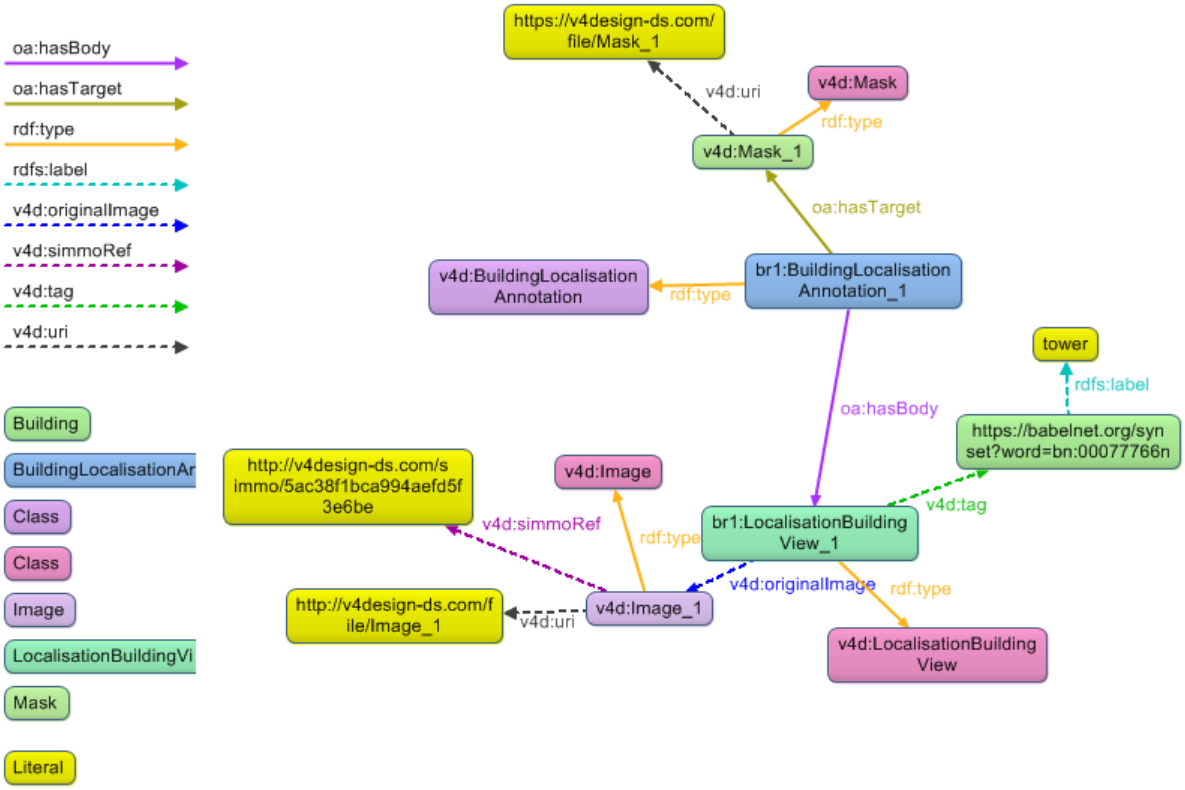

Figure 31: Example mapping of building localisation results for the simulation example

The generated knowledge graphs contains all the necessary relations to adequately map the output of building localisation. Following the annotation model described in Section 6.1, an `v4d:BuildingLocalisationAnnotation` resource is generated that is linked with the target of the annotation, i.e. the generated mask (`Mask_1`) and the annotation view (`BuildingLocalisationView_1`). The latter, defines property assertions relevant to the original image where this mask has been extracted from (`Image_1`), as well as the tag relevant tag which is the BabelNet resource for the concept “tower”. The RDF graph in the Turtle syntax[11] is given below.

```
@prefix oa: <http://www.w3.org/ns/oa#> .
@prefix v4d: <https://v4design.eu/ontologies/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

<https://v4design.eu/ontologies/simulation_v2#BuildingLocalisationAnnotation_1>
  a <https://v4design.eu/ontologies/BuildingLocalisationAnnotation> ;
  oa:hasBody <https://v4design.eu/ontologies/simulation_v2#LocalisationBuildingVie
w_1> ;
  oa:hasTarget <https://v4design.eu/ontologies/Mask_1> .

<https://v4design.eu/ontologies/Mask_1>
  a <https://v4design.eu/ontologies/Mask> ;
  v4d:uri "https://v4design-ds.com/file/Mask_1" .

v4d:simulation_v2#LocalisationBuildingView_1
  a v4d:LocalisationBuildingView ;
```

[11] https://www.w3.org/TR/turtle/

```
  v4d:originalImage v4d:Image_1 ;
  v4d:tag <https://babelnet.org/synset?word=bn:00077766n> .

<https://babelnet.org/synset?word=bn:00077766n> rdfs:label "tower" .
v4d:Image_1
  a v4d:Image ;
  v4d:simmoRef "http://v4design-ds.com/simmo/5ac38f1bca994aefd5f3e6be" ;
  v4d:uri "http://v4design-ds.com/file/Image_1" .
```

7.2 Aesthetics

The output of the aesthetics module on the image is given below.

```
{
   "simmo": "http://v4design-ds.com/simmo/<ref>",
   "image_uri": "http://v4design-ds.com/file/<imageId>",
  "tags": [
    "minimalism"
  ]
}
```

More specifically, the module generates one style (minimalism) for the building depicted in the picture. The generated knowledge graph is depicted in Figure 32. An instance of the `AestheticsAnnotation` class is defined, which is associated with the annotation target (`Image_1`) and the annotation source (instance of the `AestheticView` class). The `style` property is used to define the output of aesthetics extraction, using the BabelNet resource for minimalism.

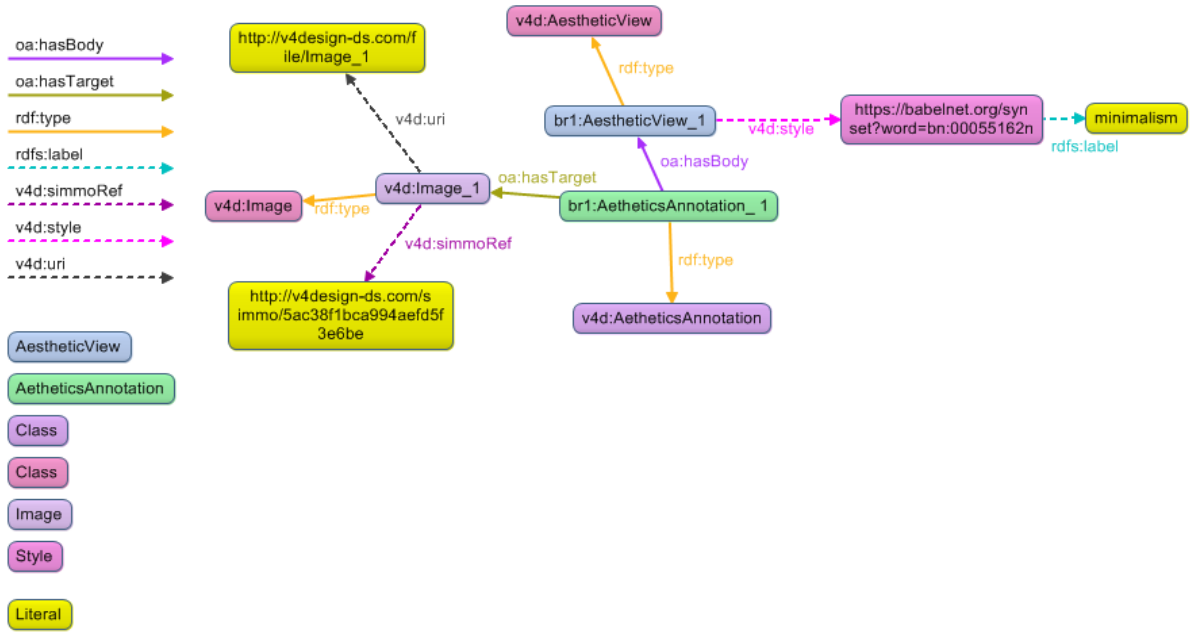

Figure 32 Example mapping of aesthetics results for the simulation example

The RDF graph in the Turtle syntax is given below.

```
@prefix oa: <http://www.w3.org/ns/oa#> .
@prefix v4d: <https://v4design.eu/ontologies/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
```

```
<https://v4design.eu/ontologies/simulation_v2#AetheticsAnnotation_1>
  a <https://v4design.eu/ontologies/AetheticsAnnotation> ;
  oa:hasBody <https://v4design.eu/ontologies/simulation_v2#AestheticView_1> ;
  oa:hasTarget <https://v4design.eu/ontologies/Image_1> .

<https://v4design.eu/ontologies/simulation_v2#AestheticView_1>
  a <https://v4design.eu/ontologies/AestheticView> ;
  v4d:style <https://babelnet.org/synset?word=bn:00055162n> .

v4d:Image_1
  a v4d:Image ;
  v4d:simmoRef "http://v4design-ds.com/simmo/5ac38f1bca994aefd5f3e6be" ;
  v4d:uri "http://v4design-ds.com/file/Image_1" .

<https://babelnet.org/synset?word=bn:00055162n> rdfs:label "minimalism" .
```

7.3 Text analysis

An excerpt of the output of text analysis on the caption of the image is give below.


```
{
  "data": {
    "simmo": "http://v4design-ds.com/simmo/<ref>"
    "dbpedia": {
      "all": [
        {
          "end": 16,
          "text": "Eiffel Tower",
          "type":
"Schema:Place,DBpedia:Place,DBpedia:ArchitecturalStructure,DBpedia:Building",
          "uri": "http://dbpedia.org/resource/Eiffel_Tower",
          "begin": 4
        },
        {
          "end": 44,
          "text": "Champ de Mars",
          "type": "",
          "uri": "http://dbpedia.org/resource/Champ_de_Mars",
          "begin": 31
        }
      ],
      "other": [
        {
          "end": 44,
          "text": "Champ de Mars",
          "type": "",
          "uri": "http://dbpedia.org/resource/Champ_de_Mars",
          "begin": 31
        }
      ]
    }
  }
}
```


The generated knowledge graph is depicted in Figure 33. More specifically, the caption of the image is represented as a `Text` media type (`Text_1`). An instance of the `TextAnalysis-Annotation` is generated for linking `Text_1` with the results of text analysis through `tag`

property assertions, which are two DBpedia concepts. It should be noted here that since the `Text_1` and the `Image_1` of the previous examples have the same `simmoRef` value, therefore we can infer that image and the text (caption in this case) belong to the same logical unit of the SIMMO model.

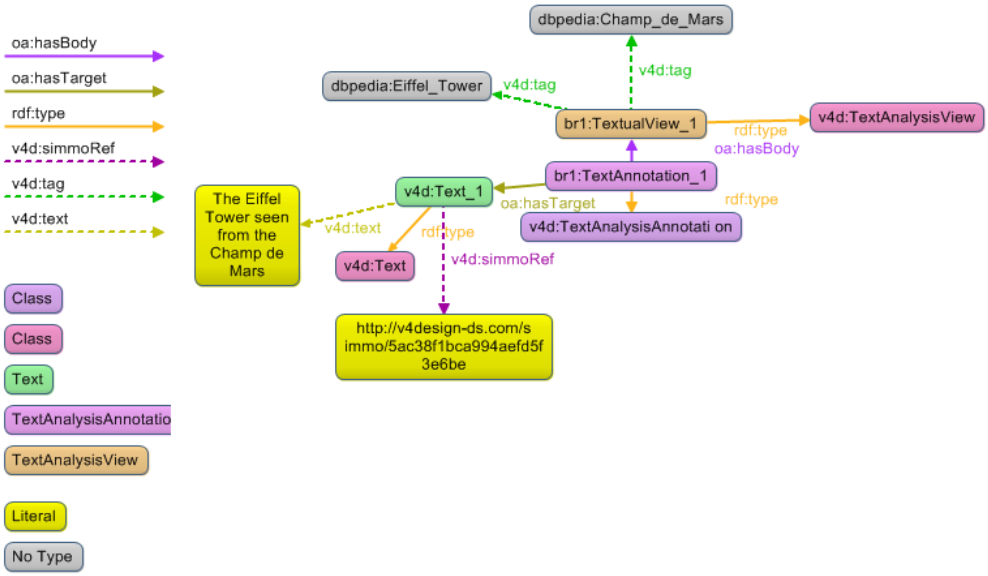

Figure 33 Example mapping of text analysis results on the caption of the image

The RDF graph in the Turtle syntax is given below.

```
@prefix oa: <http://www.w3.org/ns/oa#> .
@prefix v4d: <https://v4design.eu/ontologies/> .

<https://v4design.eu/ontologies/simulation_v2#TextAnnotation_1>
  a <https://v4design.eu/ontologies/TextAnalysisAnnotation> ;
  oa:hasBody <https://v4design.eu/ontologies/simulation_v2#TextualView_1> ;
  oa:hasTarget <https://v4design.eu/ontologies/Text_1> .

<https://v4design.eu/ontologies/simulation_v2#TextualView_1>
  a <https://v4design.eu/ontologies/TextAnalysisView> ;
  v4d:tag <http://dbpedia.org/resource/Champ_de_Mars>, <http://dbpedia.org/resourc
e/Eiffel_Tower> .

v4d:Text_1
  a v4d:Text ;
  v4d:simmoRef "http://v4design-ds.com/simmo/5ac38f1bca994aefd5f3e6be" ;
  v4d:text "The Eiffel Tower seen from the Champ de Mars" .
```

7.4 3D Model Reconstruction

3D model reconstruction needs a collection of images in order to be able to create a 3D model. In this example, we assume that the module has already provided with a collection of images, one of which is the `Image_1` we use in our example. The output of 3D model reconstruction is given below.

```
{
   "reconstructions":[
      {
         "reconstructionId":{
            "id":"1234"
         },
         "reconstructionGroupId":{
            "id":"5678"
         },
         "inputContent":[
            {
               "sourceId":"http://v4design-ds.com/file/<imageId>"
            },
            {
               "sourceId":" /3448326130_58de020bfb_o.jpg"
            }
         ],
         "usedContent":[
            {
               "sourceId":"http://v4design-ds.com/file/<imageId>"
            },
            {
               "sourceId":"/35073409352_1e142a970c_o.jpg"
            }
         ],
         "textureSize": "123456",
         "facecount": "123456"
      }
   ]
}
```


The generated knowledge graph is depicted in Figure 34. More specifically, the `_3DModelAnnotation` instance annotates the _3DModel_1 resource with the images that have been used to generate the 3D models, as well as with the two attributes relevant to the face count and texture size.

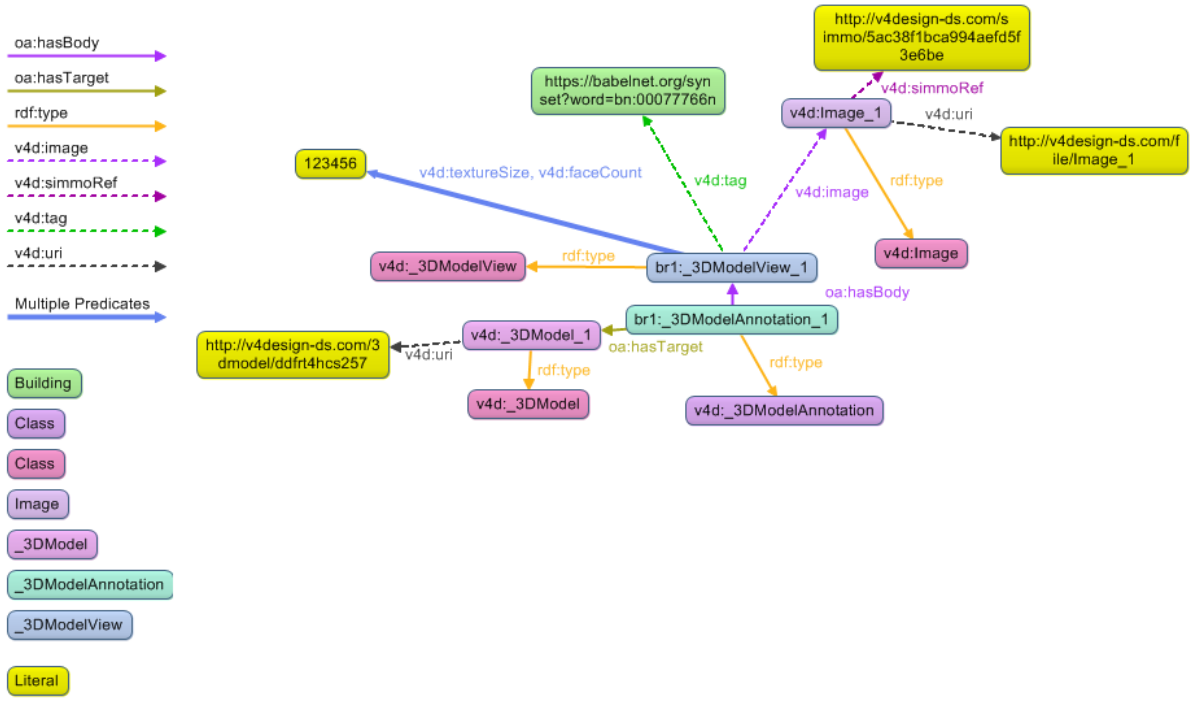

Figure 34: Example mapping of 3D model reconstruction results

The RDF graph in the Turtle syntax is given below.

```
@prefix oa: <http://www.w3.org/ns/oa#> .
@prefix v4d: <https://v4design.eu/ontologies/> .

<https://v4design.eu/ontologies/simulation_v2#_3DModelAnnotation_1>
  a <https://v4design.eu/ontologies/_3DModelAnnotation> ;
  oa:hasBody <https://v4design.eu/ontologies/simulation_v2#_3DModelView_1> ;
  oa:hasTarget <https://v4design.eu/ontologies/_3DModel_1> .

<https://v4design.eu/ontologies/simulation_v2#_3DModelView_1>
  a <https://v4design.eu/ontologies/_3DModelView> ;
  v4d:faceCount "123456" ;
  v4d:textureSize "123456" ;
  v4d:image v4d:Image_1 ;
  v4d:tag <https://babelnet.org/synset?word=bn:00077766n> .

v4d:_3DModel_1
  a v4d:_3DModel ;
  v4d:uri "http://v4design-ds.com/3dmodel/ddfrt4hcs257" .

v4d:Image_1
  a v4d:Image ;
  v4d:simmoRef "http://v4design-ds.com/simmo/5ac38f1bca994aefd5f3e6be" ;
  v4d:uri "http://v4design-ds.com/file/Image_1" .
```

7.5 Language Generation

An example of the output of text generation is given below.

```
{
   "simmo": "http://v4design-ds.com/simmo/<ref>",
   "text": "this is an example summary",
   "lang": "en"
}
```

The generated knowledge graphs are shown in Figure 35. The instance of the `TextGenerationAnnotation` defines a view directly on the 3D model instance, describing the textual description that should be presented in the user. In this example, we just illustrate of the 3D model with an example text (also, the `lang` attribute is not depicted).

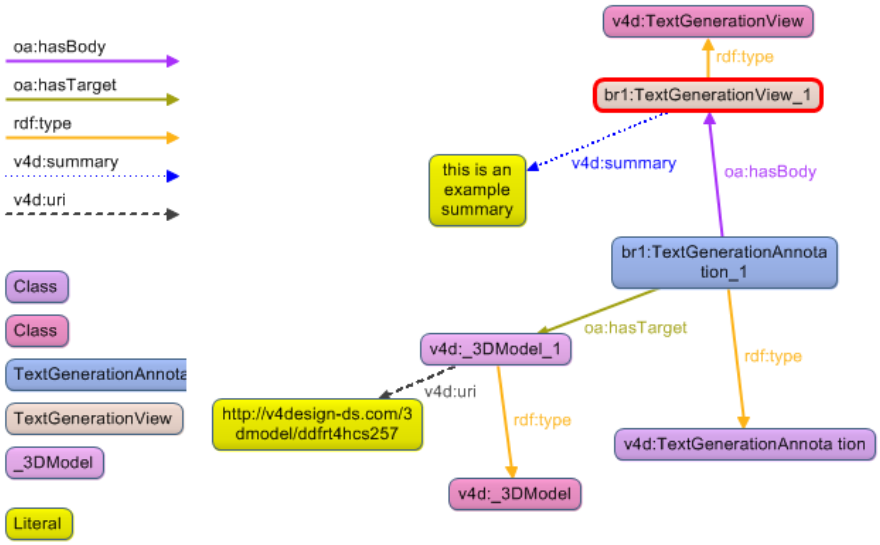

Figure 35: Example mapping of text generation results

The RDF graph in the Turtle syntax is given below.

```
@prefix ns0: <http://www.w3.org/ns/oa#> .
@prefix ns1: <https://v4design.eu/ontologies/> .

<https://v4design.eu/ontologies/simulation_v2#TextGenerationAnnotation_1>
  a <https://v4design.eu/ontologies/TextGenerationAnnotation> ;
  ns0:hasBody <https://v4design.eu/ontologies/simulation_v2#TextGenerationView_1>
;
  ns0:hasTarget <https://v4design.eu/ontologies/_3DModel_1> .

<https://v4design.eu/ontologies/_3DModel_1>
  a <https://v4design.eu/ontologies/_3DModel> ;
  ns1:uri "http://v4design-ds.com/3dmodel/ddfrt4hcs257" .

ns1:simulation_v2#TextGenerationView_1
  a ns1:TextGenerationView ;
  ns1:summary "this is an example summary" .
```

Figure 36 presents the complete RDF knowledge graph that is generated for the simulation example.

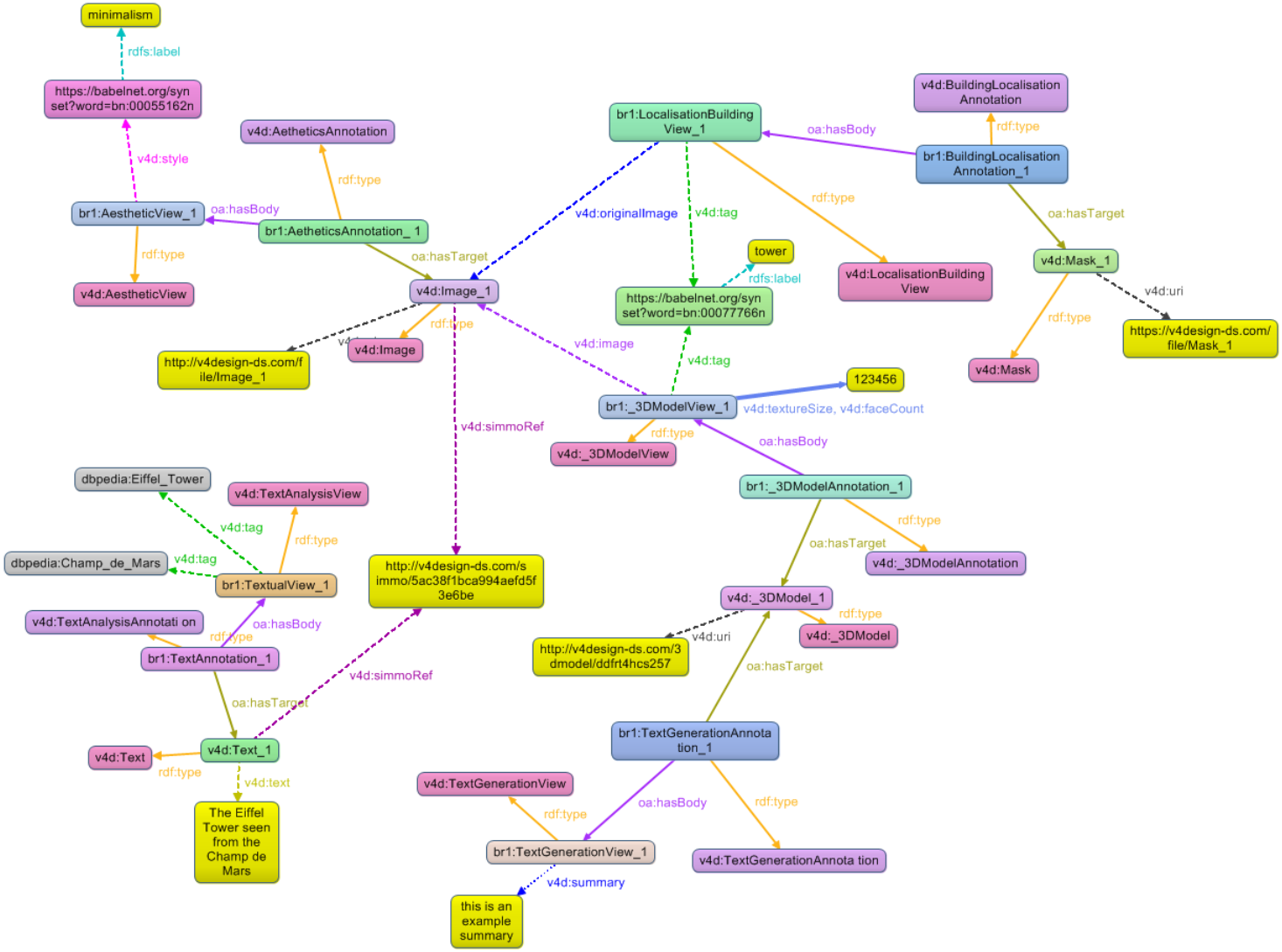

Figure 36: The complete RDF knowledge graph with the annotation model of the simulation example

8 CONCLUSIONS

In this document we provided the requirement specifications and the state-of-the-art analysis relevant to the building of the semantic knowledge structures addressed within "***T5.1: Semantic content representation"***. We also described the current status of the V4Design ontologies towards MS2 that encode in a structured way the vocabulary and the precise semantics of information relevant to the V4Design application context. We have also presented the preliminary version of WP5's reasoning framework towards MS2 ***("T5.2: Semantic integration and reasoning"***) for combining, integrating and semantically interpreting and enriching the knowledge captured in the KB. The current annotation model of V4Design has been validated through a simulation example organised within WP5 in order to elicit modelling requirements and acquire a better understanding of the structure and content of the outputs provided by each component of the V4Design pipeline.

Next steps include further enrichments and enhancements of WP5 ontology-based framework in three main directions. First, to refine the already developed annotation models and to provide and validate additional ontology constructs for capturing richer domain knowledge pertinent population of the KB with data (e.g. BIM), based on the richer output the various modules will provide towards the first prototype (M18). The annotation model will be also enriched with additional metadata properties, when it is a clear view on the exact output of the analysis, e.g. the annotations of the 3D models. Second, to enhance the reasoning capabilities that will address more elaborate interpretation aspects by (i) enriching the supported semantics both at the terminological level, by defining additional class and property axioms, and at the assertional level by incorporating inference rules, (ii) handling imperfect information (i.e. missing or uncertain inputs). Special emphasis will be also place on aggregating the results of textual analysis for entity disambiguation. Finally, in parallel with "***T5.3: Linked data for dynamic 3D objects retrieval"***, efficient searching mechanisms will be implemented in order to provide an intelligent query interface for addressing users' searching requirements.

9 REFERENCES

Baader, Franz. et al. 2003. Description Logic Handbook *The Description Logic Handbook: Theory, Implementation, and Applications*. eds. Franz Baader et al. Cambridge University Press. http://dl.acm.org/citation.cfm?id=885746 (September 14, 2015).

Breslin, John G., Andreas Harth, Uldis Bojars, and Stefan Decker. 2005. "Towards Semantically-Interlinked Online Communities." *The Semantic Web: Research and Applications*.

Ciccarese, Paolo et al. 2013. "PAV Ontology: Provenance, Authoring and Versioning." *Journal of Biomedical Semantics* 4(1): 37. http://jbiomedsem.biomedcentral.com/articles/10.1186/2041-1480-4-37 (October 15, 2018).

Deborah L. McGuinness, Frank van Harmelen. 2004. "Owl Web Ontology Language Overview." *W3C recommendation 10.2004-03*.

Doerr, Martin et al. 2010. "The Europeana Data Model (EDM)." In *World Library and Information Congress*,.

Fernández-López, M, A Gómez-Pérez, and Natalia Juristo. 1997. "METHONTOLOGY: From Ontological Art Towards Ontological Engineering." *AAAI-97 Spring Symposium Series*.

Gangemi, Aldo, and Peter Mika. 2003. "Understanding the Semantic Web through Descriptions and Situations." In *Proceedings of ODBASE03 Conference*, Springer, Berlin, Heidelberg, 689–706. http://link.springer.com/10.1007/978-3-540-39964-3_44 (July 12, 2017).

Glimm, Birte et al. 2014. "HermiT: An OWL 2 Reasoner." *Journal of Automated Reasoning*.

Grau, Bernardo Cuenca et al. 2008. "OWL 2: The next Step for OWL." *Web Semantics: Science, Services and Agents on the World Wide Web* 6(4): 309–22. http://linkinghub.elsevier.com/retrieve/pii/S1570826808000413 (June 23, 2017).

Grosof, Benjamin N, Ian R Horrocks, Raphael Volz, and Stefan Decker. 2003. "Description Logic Programs: Combining Logic Programs with Description Logic." *Proceedings of the 12th international conference on World Wide Web*.

Gruber, Thomas R. 1993. "A Translation Approach to Portable Ontology Specifications." *Knowledge Acquisition*.

Haarslev, Volker, and Ralf Möller. 2003. "Racer: A Core Inference Engine for the Semantic Web." *Proceedings of the 2nd International Workshop on Evaluation of Ontologybased Tools*.

Van Hage, Willem Robert et al. 2011. "Design and Use of the Simple Event Model (SEM)." *Journal of Web Semantics*.

Harris, Steve, and Andy Seaborne. 2013. W3C Recommendation *SPARQL 1.1 Query Language*.

He, S. Y. et al. 2004. "Effects of Pres0sure Reduction Rate on Quality and Ultrastructure of Iceberg Lettuce after Vacuum Cooling and Storage." In *Postharvest Biology and*

Technology, , 263–73.

Horrocks, Ian et al. 2004. “SWRL : A Semantic Web Rule Language Combining OWL and RuleML.” *W3C Member submission 21*.

ter Horst, Herman J. 2004. “Extending the RDFS Entailment Lemma.” In *The Semantic Web – ISWC 2004*,.

Kalogerakis, Evangelos, Stavros Christodoulakis, and Nektarios Moumoutzis. 2006. “Coupling Ontologies with Graphics Content for Knowledge Driven Visualization.” In *Proceedings - IEEE Virtual Reality*,.

Knublauch, Holger, James A. Hendler, and Kingsley Idehen. 2011. “SPIN - Overview and Motivation W3C: Member Submission 22 February 2011.” *22 February 2011*.

Lebo, Timothy, Satya Sahoo, and D McGuinness. 2013. “PROV-O: The PROV Ontology.” *W3C Recommendation*.

Miles, Alistair. 2006. “SKOS Core Vocabulary Specification.” *English* (November 2005): 1–28. https://www.w3.org/TR/swbp-skos-core-spec/ (October 15, 2018).

Motik, Boris, Ulrike Sattler, and Rudi Studer. 2005. “Query Answering for OWL-DL with Rules.” *Web Semantics*.

De Nicola, Antonio, Michele Missikoff, and Roberto Navigli. 2005. “A Proposal for a Unified Process for Ontology Building: UPON.” In Springer, Berlin, Heidelberg, 655–64. http://link.springer.com/10.1007/11546924_64 (October 15, 2018).

Noy, Natalya F., and Deborah L. McGuinness. 2001. “Ontology Development 101: A Guide to Creating Your First Ontology.” *Stanford Knowledge Systems Laboratory*.

Passant, Alexandre, Uldis Bojars, John G. Breslin, and Stefan Decker. 2010. “The SIOC Project: Semantically-Interlinked Online Communities, from Humans to Machines.” In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*,.

Perez, Jorge, Marcelo Arenas, and Claudio Gutierrez. 2006. “Semantics and Complexity of SPARQL.” : 30–43. http://link.springer.com/10.1007/11926078_3 (October 15, 2018).

Rosati, Riccardo. 2006. “DL+log: Tight Integration of Description Logics and Disjunctive Datalog.” In *The Tenth International Conference on Principles of Knowledge Representation and Reasoning KR2006*,.

Sanderson, Robert, Paolo Ciccarese, and Benjamin Young. 2017. W3C *Web Annotation Data Model*. https://www.w3.org/TR/annotation-model/ (October 15, 2018).

Scherp, Ansgar, Thomas Franz, Carsten Saathoff, and Steffen Staab. 2009. “{F--A} Model of Events Based on the Foundational Ontology {Dolce+DnS Ultralight}.” In *Fifth International Conf. on Knowledge Capture*, , 137–44. http://doi.acm.org/10.1145/1597735.1597760.

Sikos, L. F. 2017. “3D Model Indexing in Videos for Content-Based Retrieval via X3D-Based Semantic Enrichment and Automated Reasoning.” In *Proceedings of the 22nd International Conference on 3D Web Technology - Web3D ’17*,.

Sikos, L.F. 2015. Mastering Structured Data on the Semantic Web: From HTML5 Microdata to

Linked Open Data *Mastering Structured Data on the Semantic Web: From HTML5 Microdata to Linked Open Data*.

Sikos, Leslie F, and David M W Powers. 2015. “Knowledge-Driven Video Information Retrieval with LOD: From Semi-Structured to Structured Video Metadata.” *Proceedings of the Eighth Workshop on Exploiting Semantic Annotations in Information Retrieval*.

Sirin, Evren et al. 2007. “Pellet: A Practical OWL-DL Reasoner.” *Web Semantics*.

Staab, Steffen, Rudi Studer, Hans Peter Schnurr, and York Sure. 2001. “Knowledge Processes and Ontologies.” *IEEE Intelligent Systems and Their Applications*.

Stegmaier, Florian et al. 2013. “Unified Access to Media Metadata on the Web.” *IEEE Multimedia* 20(2): 22–29. http://ieeexplore.ieee.org/document/6353418/ (October 15, 2018).

Studer, Rudi, V.Richard Benjamins, and Dieter Fensel. 1998. “Knowledge Engineering: Principles and Methods.” *Data & Knowledge Engineering* 25(1–2): 161–97. https://www.sciencedirect.com/science/article/pii/S0169023X97000566 (May 4, 2018).

Suárez-Figueroa, Mari Carmen, Asunción Gómez-Pérez, and Boris Villazón-Terrazas. 2009. “How to Write and Use the Ontology Requirements Specification Document.” In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*,.

Tsarkov, Dmitry, and Ian Horrocks. 2006. “FaCT++ Description Logic Reasoner: System Description.” In Springer, Berlin, Heidelberg, 292–97. http://link.springer.com/10.1007/11814771_26 (October 15, 2018).

Vardi, Moshe Y. 1996. “Why Is Modal Logic so Robustly Decidable?” In *Descriptive Complexity and Finite Models: Proceedings of a DIMACS Workshop*,.

Vasilakis, George et al. 2007. “A Common Ontology for Multi-Dimensional Shapes.” https://core.ac.uk/download/pdf/37832944.pdf (October 15, 2018).

A. Appendix

A.1. Simulation example RDF annotation graph

```
@prefix oa: <http://www.w3.org/ns/oa#> .
@prefix v4d: <https://v4design.eu/ontologies/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

<https://v4design.eu/ontologies/simulation_v2#BuildingLocalisationAnnotation_1>
  a <https://v4design.eu/ontologies/BuildingLocalisationAnnotation> ;
  oa:hasBody
<https://v4design.eu/ontologies/simulation_v2#LocalisationBuildingView_1> ;
  oa:hasTarget <https://v4design.eu/ontologies/Mask_1> .

<https://v4design.eu/ontologies/simulation_v2#LocalisationBuildingView_1>
  a <https://v4design.eu/ontologies/LocalisationBuildingView> ;
  v4d:originalImage v4d:Image_1 ;
  v4d:tag <https://babelnet.org/synset?word=bn:00077766n> .

<https://babelnet.org/synset?word=bn:00077766n> rdfs:label "tower" .
v4d:Image_1
  a v4d:Image ;
  v4d:simmoRef "http://v4design-ds.com/simmo/5ac38f1bca994aefd5f3e6be" ;
  v4d:uri "http://v4design-ds.com/file/Image_1" .

v4d:simulation_v2#AetheticsAnnotation_1
  oa:hasTarget v4d:Image_1 ;
  a v4d:AetheticsAnnotation ;
  oa:hasBody v4d:simulation_v2#AestheticView_1 .

v4d:simulation_v2#_3DModelView_1
  v4d:image v4d:Image_1 ;
  a v4d:_3DModelView ;
  v4d:faceCount "123456" ;
  v4d:textureSize "123456" ;
  v4d:tag <https://babelnet.org/synset?word=bn:00077766n> .

v4d:simulation_v2#AestheticView_1
  a v4d:AestheticView ;
  v4d:style <https://babelnet.org/synset?word=bn:00055162n> .

<https://babelnet.org/synset?word=bn:00055162n> rdfs:label "minimalism" .
v4d:simulation_v2#_3DModelAnnotation_1
  oa:hasBody v4d:simulation_v2#_3DModelView_1 ;
  a v4d:_3DModelAnnotation ;
  oa:hasTarget v4d:_3DModel_1 .

v4d:_3DModel_1
  a v4d:_3DModel ;
  v4d:uri "http://v4design-ds.com/3dmodel/ddfrt4hcs257" .

v4d:simulation_v2#TextGenerationAnnotation_1
  oa:hasTarget v4d:_3DModel_1 ;
  a v4d:TextGenerationAnnotation ;
  oa:hasBody v4d:simulation_v2#TextGenerationView_1 .

v4d:simulation_v2#TextGenerationView_1
```

```
  a v4d:TextGenerationView ;
  v4d:summary "this is an example summary" .

v4d:Mask_1
  a v4d:Mask ;
  v4d:uri "https://v4design-ds.com/file/Mask_1" .

v4d:simulation_v2#TextAnnotation_1
  a v4d:TextAnalysisAnnotation ;
  oa:hasBody v4d:simulation_v2#TextualView_1 ;
  oa:hasTarget v4d:Text_1 .

v4d:Text_1
  a v4d:Text ;
  v4d:simmoRef "http://v4design-ds.com/simmo/5ac38f1bca994aefd5f3e6be" ;
  v4d:text "The Eiffel Tower seen from the Champ de Mars" .

v4d:simulation_v2#TextualView_1
  a v4d:TextAnalysisView ;
  v4d:tag <http://dbpedia.org/resource/Champ_de_Mars>,
<http://dbpedia.org/resource/Eiffel_Tower> .
```

A.2. Vocabulary mappings

skos	<http://www.w3.org/2004/02/skos/core>
oa	<http://www.w3.org/ns/oa>
v4d	<https://v4design.eu/ontologies>
edm	<http://www.europeana.eu/schemas/edm>
vidont	<http://vidont.org>
ore	<http://www.openarchives.org/ore/terms/>
x3d	<http://purl.org/ontology/x3d>
schema	<http://schema.org/>

V4Design concept	SKOS relation	External concept
Classes		
v4d:_3DModel	skos:exactMatch	x3d:3DModel
v4d:_3DModelAnnotation	skos:broadMatch skos:relatedMatch	oa:Annotation ore:Aggregation
v4d:AestheticsAnnotation	skos:broadMatch skos:relatedMatch	oa:Annotation ore:Aggregation
v4d:Building	skos:broader	schema:CivicStructure
v4d:BuildingLocalisationAnnotation	skos:broadMatch skos:relatedMatch	oa:Annotation ore:Aggregation
v4d:Creator	skos:relatedMatch	edm:Agent
v4d:Image	skos:exactMatch	schema:ImageObject
v4d:Mask	skos:broader	schema:ImageObject

v4d:MediaType	exact:match	schema:MediaObject
v4d:ObjectLocalisationAnnotation	skos:broadMatch skos:relatedMatch	oa:Annotation ore:Aggregation
v4d:Text	skos:exactMatch	schema:Text
v4d:TextAnalysisAnnotation	skos:broadMatch skos:relatedMatch	oa:Annotation ore:Aggregation
v4d:TextGenerationAnnotation	skos:broadMatch skos:relatedMatch	oa:Annotation ore:Aggregation
v4d:Texture	skos:exactMatch	x3d:Texture
v4d:Video	skos:exactMatch	schema:VideoObject vidont:Video
v4d:View	skos:relatedMatch	ore:Proxy
Properties		
creator	skos:exactMatch	schema:creator
image	skos:exactMatch	schema:image x3d:image
text	skos:exactMatch	schema:text
uri	skos:exactMatch	dcterms:identifier

A.3. Ontologies

The current version of the V4Design annotation model (described in Section 6.1) can be found at: https://v4design.eu/wp-content/uploads/2018/10/v4design_ontologies_v1.zip.

The archive also contains the instantiation of the model for the simulation example described in Section 7.