



beAWARE

Enhancing decision support and management services in extreme weather
climate events

700475

D4.2

Semantic Representation and Preliminary Report on Reasoning

Dissemination level:	Public
Contractual date of delivery:	Month 18, 30 June 2018
Actual date of delivery:	Month 18, 30 June 2018
Workpackage:	WP4: Aggregation and semantic integration of emergency information for decision support and early warnings generation
Task:	T4.3 - Semantic representation T4.5 -Reasoning for decision support
Type:	Report
Approval Status:	Final
Version:	1.0
Number of pages:	80
Filename:	D4.2_beAWARE_Semantic_Representation_and_Reasoning_2018-06-30_v1.0

Abstract



This deliverable presents the first iteration of the beAWARE knowledge base, which is a knowledge representation model for semantically representing notions pertinent to the project. Two additional core components are presented: (a) the Knowledge Base Repository (KBR) for storing the ontology, and, (b) the Knowledge Base Service (KBS), which serves as the interface to the ontology. The document also presents our preliminary results on semantic reasoning (running on top of the ontology as part of the KBS), which will be further refined according to the needs that will emerge from the upcoming pilots.

The information in this document reflects only the author's views and the European Community is not liable for any use that may be made of the information contained therein. The information in this document is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.



Co-funded by the European Union



This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 700475

History

Version	Date	Description / Reason of change	Revised by
0.1	03.04.2018	Deliverable outline	CERTH, IOSB
0.2	11.05.2018	Contributions in Chapters 1, 2, Appendix A	CERTH
0.3	22.05.2018	Contributions in Chapters 2, 4, finished Appendix A	CERTH
0.4	29.05.2018	Contributions in Chapters 2, 3, 4, Appendix B	CERTH, IOSB
0.5	04.06.2018	Final contributions in the whole document	CERTH, IOSB
0.6	06.06.2018	Document submitted for internal review	CERTH
0.7	28.06.2018	Addressed internal review remarks	CERTH, MSIL
1.0	29.06.2018	Final version for submission	CERTH

Author List

Organisation	Name	Contact Information
CERTH	E. Kontopoulos	skontopo@iti.gr
	S. Vrochidis	stefanos@iti.gr
	A. Karakostas	akarakos@iti.gr
	I. Kompatsiaris	ikom@iti.gr
IOSB	J. Moßgraber	juergen.mossgraber@iosb.fraunhofer.de
	T. Hellmund	tobias.hellmund@iosb.fraunhofer.de
	H. van der Schaaf	hylke.vanderschaaf@iosb.fraunhofer.de

Reviewers

Organisation	Name	Contact Information
MSIL	Y. Mordecai	yaniv.mordecai@motorolasolutions.com

Executive Summary

This document constitutes deliverable D4.2 “*Semantic Representation and Preliminary Report on Reasoning*” and focuses on presenting **the first iteration of the beAWARE ontology**. The latter, also referred to as “**the beAWARE Knowledge Base (KB)**”, is a knowledge representation model for semantically representing notions pertinent to the project: (a) climate-related natural disasters, (b) analysis of data from the multimodal sensors, and, (c) rescue team assignments. The beAWARE KB is a central component in the system architecture, and is hosted by the **Knowledge Base Repository (KBR)**, while other modules repeatedly interact with the KB via the **Knowledge Base Service (KBS)**, which serves as the interface to the ontology. The KBS accepts input from other modules and semantically integrates it into the ontology. It also receives output from the semantic reasoning process running on top of the KB and forwards the inferred, high-level knowledge back to other interested system modules, like the report generator and the Public Safety Answering Point (PSAP). Figure 1 depicts the interaction between these components.

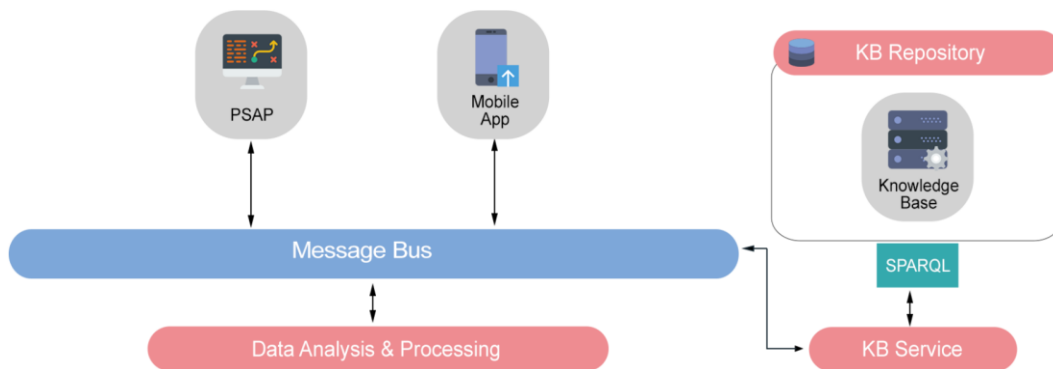


Figure 1. Interaction between the KB, KBS, KBR and other key beAWARE components.

The above components are presented in more detail in the rest of the document, and, more specifically:

- **Chapter 0** presents all the relevant background notions, like ontologies and the Semantic Web;
- **Chapter 2** reports on the design, implementation and evaluation of the first version of the beAWARE ontology;
- **Chapter Error! Reference source not found.** presents WebGenesis, the system that hosts the beAWARE KB, which is also a project output and serves as the KBR;
- **Chapter 4** presents the KBS component, which is responsible for the semantic integration and reasoning – preliminary results on the reasoning component are discussed, which will be further extended after the execution of the pilots in the coming months;
- Finally, **Chapter 5** concludes with closing remarks and directions for improving the ontology and all the accompanying tools and mechanisms towards the final version.

Abbreviations and Acronyms

The following abbreviations have been used in this document:

CMS	<i>Content Management System</i>
CQ	<i>Competency Question</i>
DL	<i>Description Logics</i>
KB	<i>Knowledge Base</i>
KBR	<i>Knowledge Base Repository</i>
KBS	<i>Knowledge Base Service</i>
ODP	<i>Ontology Design Pattern</i>
OWL	<i>Web Ontology Language</i>
PSAP	<i>Public Safety Answering Point</i>
W3C	<i>World Wide Web Consortium</i>
WWW	<i>World Wide Web</i>

Table of Contents

EXECUTIVE SUMMARY	4
ABBREVIATIONS AND ACRONYMS.....	5
TABLE OF CONTENTS.....	6
LIST OF FIGURES.....	11
LIST OF TABLES	11
1 BACKGROUND	12
1.1 Ontologies in the Semantic Web	12
1.2 Crisis Management Ontologies	12
1.3 Ontology Engineering	14
1.4 Chapter Summary	15
2 THE BEAWARE KB V1.....	16
2.1 Specification of Ontology Requirements	16
2.2 Reuse of Existing Resources	17
2.3 Ontology Conceptualization.....	18
2.3.1 Representing Natural Disasters	18
2.3.2 Representing Analyzed Data.....	20
2.3.3 Representing Rescue Team Assignments	20
2.4 Semantic SensorThings Mapping	21
2.4.1 Base Model	21
2.4.2 MultiDataStream	23
2.4.3 Actuation	23
2.4.4 Mapping the Sensor Model to the beAWARE Ontology	24
2.5 Ontology Formalization and Implementation	24
2.6 Ontology Evaluation	25
2.6.1 Evaluating the Consistency and Quality	25
2.6.2 Evaluating the Structure	26
2.6.3 Compliance with User Requirements	27
2.7 Chapter Summary	27
3 KNOWLEDGE BASE REPOSITORY.....	28
3.1 WebGenesis Characteristics and Scalability	28
3.2 Navigating the beAWARE Knowledge Base.....	28
3.3 Editing an Ontology Image.....	30
3.4 Populating the Ontology.....	31
3.5 Requesting Data from the Ontology.....	33
3.6 User Management.....	35
3.7 Chapter Summary	35
4 SEMANTIC INTEGRATION AND REASONING	36

4.1	Semantic Integration	36
4.2	Semantic Reasoning	41
4.2.1	Calculation of Incident Severity Index	41
4.2.2	Spatial Clustering of Incidents	42
4.2.3	Monitoring of Safe Locations.....	42
4.3	Chapter Summary	43
5	CONCLUSIONS AND NEXT STEPS.....	44
	REFERENCES	46
	APPENDIX A – ONTOLOGY SPECIFICATION.....	49
	Classes	49
	Agriculture	49
	Animal.....	49
	Asset	49
	Audio Item	49
	Bridge.....	49
	Building	49
	Car.....	49
	Climate Parameter.....	50
	Climate Parameter Type	50
	Communication	50
	Data Analysis.....	50
	Dataset.....	50
	Detection	50
	Dunes	50
	Ecological Asset	50
	Educational Facility.....	51
	Electric Energy Supply.....	51
	Energy	51
	Fire Department	51
	Garbage Collection	51
	Hospital.....	51
	Human	51
	Image Analysis	51
	Image Item.....	52
	Impact.....	52
	Impact Type	52
	Incident.....	52
	Incident Type	52
	Infrastructure.....	52
	Levee.....	52
	Livestock	52

Living Being	53
Location	53
Media Item	53
Mission.....	53
Monument.....	53
Natural Disaster	53
Natural Disaster Type	53
Natural Habitat	54
Plant.....	54
Police	54
Property.....	54
Public Transportation	54
Responder.....	54
River.....	54
Sensor	54
Sewer	55
Square.....	55
Street	55
Structure	55
Subway.....	55
Task.....	55
Text Analysis	55
Text Item.....	55
Transportation	56
Vehicle	56
Video Analysis.....	56
Video Item	56
Vulnerable Object.....	56
Wall.....	56
Water Supply	56
Wildlife.....	56
Object Properties	57
caused by disaster	57
caused by incident.....	57
causes disaster impact.....	57
causes incident impact	57
characterized by parameter type	57
characterizes disaster type	58
contained in dataset.....	58
contains detection	58
derived from	58
detects incident	58

detects participant.....	59
has climate parameter measurement	59
has climate parameter occurrence.....	59
has disaster location	59
has disaster occurrence	59
has impact occurrence.....	60
has impact on	60
has incident climate parameter.....	60
has incident impact.....	60
has incident location.....	60
has incident occurrence.....	61
has measurement location	61
has media location.....	61
has related incident	61
has report location	61
has responder location	62
incident detected by.....	62
involved in dataset	62
involved in incident.....	62
involves incident	62
involves participant	63
is assigned mission.....	63
is assigned to responder.....	63
is climate parameter of incident.....	63
is impact of incident	63
is impacted by	64
is location of disaster	64
is location of incident	64
is location of measurement.....	64
is location of media item	64
is location of report	65
is location of responder	65
is of climate parameter type	65
is of disaster type.....	65
is of impact type	65
is of incident type	66
is result of	66
leads to	66
observes.....	66
participant detected by	66
produced by task	67
relates to.....	67

relates to incident.....	67
relates to media item	67
relates to mission	67
relates to natural disaster.....	68
relates to task	68
sensor produces dataset.....	68
task produces dataset.....	68
Data Properties	68
belongs to collection.....	68
has analyzed media source	69
has dataset results source	69
has detection confidence	69
has detection end	69
has detection risk.....	69
has detection start.....	70
has detection timestamp.....	70
has incident end	70
has incident priority.....	70
has incident start	70
has incident severity.....	71
has latitude	71
has longitude	71
has measurement timestamp.....	71
has media item timestamp	71
has mission end	71
has mission priority	72
has mission start	72
has mission status.....	72
has radius.....	72
has raw media source.....	72
has report ID	72
has unit	73
has value	73
APPENDIX B – COMPETENCY QUESTIONS AS SPARQL QUERIES	74

List of Figures

Figure 1. Interaction between the KB, KBS, KBR and other key beAWARE components.	4
Figure 2. High-level overview of the core classes of the beAWARE ontology.	18
Figure 3. Representation of climate-related natural disasters in the beAWARE ontology.	19
Figure 4. Instantiation of a sample temperature measurement in the beAWARE ontology.	19
Figure 5. Representation of analyzed data in the beAWARE ontology.	20
Figure 6. A video analysis example in the beAWARE ontology.	20
Figure 7. Representation of mission assignments to first responder units in the beAWARE ontology.	21
Figure 8. The OGC SensorThings API data model.	22
Figure 9. The entry point of the beAWARE ontology in WebGenesis.	29
Figure 10. Displaying a class in WebGenesis.	29
Figure 11. Viewing an instance in WebGenesis.	30
Figure 12. Editing ontology images in WebGenesis.	31
Figure 13. Creating an input form in WebGenesis.	31
Figure 14. The WebGenesis SPARQL-query editor.	34
Figure 15. KBS connectivity diagram.	36

List of Tables

Table 1. Comparative representation aspects of crisis management ontologies.	13
Table 2. Competency Questions (CQs) that drove the design of the beAWARE ontology v1.	16
Table 3. Mapping of the SensorThings API entities to the beAWARE ontology concepts.	24
Table 4. Ontology metrics for the beAWARE ontology v1, generated by OntoMetrics.	26

1 Background

In order to facilitate the understanding, sharing and reuse of knowledge between different systems, it is essential to define common vocabularies that represent shared knowledge in a formal way. **Ontologies** constitute the specification of a vocabulary for semantically representing a shared domain of discourse [Gruber, 1993]. An ontology semantically models knowledge by defining a set of **classes** (objects, concepts, and other entities) existing in some domain of interest, and their **properties** (attributes, i.e. relationships that hold between them). The expressiveness of the ontology depends on the knowledge representation language used.

This chapter presents the relevant background notions. Starting with the role of the ontologies in the Semantic Web, we then give an overview of existing ontology-based models for representing domains pertinent to the beAWARE project. Finally, we justify the ontology engineering approach we followed for designing and developing the beAWARE ontology, before delving into the detailed presentation of the ontology in Chapter 2.

1.1 Ontologies in the Semantic Web

The **Semantic Web** is "*a web of data that can be processed directly and indirectly by machines*" [Berners-Lee et al., 2001]. It is an extension of the World Wide Web (WWW), in which web resources are augmented with semantics describing their intended meaning in a formal, machine-understandable way. The term was coined by Tim Berners-Lee, the inventor of WWW and director of the World Wide Web Consortium (W3C), which oversees the development of proposed Semantic Web standards. The standards proposed by W3C promote common data formats and exchange protocols on the Web. The Semantic Web is thus regarded as an integrator across different content, information applications, and systems.

Ontologies play a key role in the Semantic Web, providing the machine-interpretable semantic vocabulary and serving as the knowledge representation and exchange vehicle. The **Web Ontology Language (OWL)** has emerged as the official W3C recommendation for creating and sharing ontologies on the Web [Bechhofer, 2009].

1.2 Crisis Management Ontologies

The emergence of Semantic Web technologies [Hendler, 2009] has led to the widespread adoption of ontology-based approaches in various domains, including crisis management. A recent thorough review of the state of the art in crisis management ontologies is given in [Liu et al., 2013], while two of the most prominent approaches in crisis management and response are **MOAC** (Management of a Crisis) [Limbu, 2012] and **SoKNOS** [Babitski et al., 2011]. MOAC is a lightweight vocabulary that provides terms for linking crisis information

from three different sources: (a) traditional humanitarian agencies, (b) volunteer and technical committees and (c) disaster affected communities. The vocabulary has been developed based on contributions from various key stakeholders, like the Inter Agency Standing Committee (IASC), the Global Shelter Cluster, and the Ushahidi platform, who were also involved in assessing MOAC's usability, functionality, and structure.

SoKNOS, on the other hand, is a set of ontologies ensuring that newly created information, as well as integrated sensor information, is semantically characterized, supporting the goal of a shared and semantically unambiguous information basis across organizations managing crisis incidents. The central SoKNOS ontology is a core domain ontology defining the basic vocabulary of the emergency management domain. Additional dedicated ontologies are used for representing resources and damages, and deployment regulations defining the relations between resources and damages. Furthermore, for the definition of system components, ontologies of user interfaces and interactions as well as geo sensors have been developed. Based on the aforementioned ontologies, additional specialized application ontologies can be defined for each application used in the disaster scenario. All SoKNOS ontologies have been developed in close cooperation with domain experts.

Besides the above, other prominent approaches proposed in literature include SOFERS [Liu et al., 2014], ISyCri [Truptil et al., 2008], and the ontologies proposed by Lauras et al. (2015), Mescherin et al. (2013), and Zavarella et al. (2014).

Table 1. Comparative representation aspects of crisis management ontologies.

Approach	<i>Disasters/crises/emergencies</i>	<i>Climate parameters</i>	<i>Sensor data analysis</i>	<i>Response units & equipment</i>	<i>Response unit assignments</i>	<i>Incidents</i>	<i>Impacts</i>
BACAREX (2005)	<i>Only fire</i>	☑		☑	☑		
ISyCri (2008)	<i>Minimal</i>					☑	☑
SoKNOS (2011)				☑	☑	☑	☑
MOAC (2012)	☑					☑	☑
Mescherin et al. (2013)	<i>Minimal</i>			☑	☑	<i>Minimal</i>	<i>Minimal</i>
SOFERS (2014)	☑	☑		☑		☑	☑
Zavarella et al. (2014)	☑						☑
Lauras et al. (2015)	☑			☑	☑	☑	☑
beAWARE v1 (2018)	☑	☑	☑	☑	☑	☑	☑

Finally, another highly relevant approach, albeit rather outdated, is the BACAREX ontology [de la Asunción et al., 2005], which is part of the SIADEx framework for facilitating the design of plans for fighting forest fires. More specifically, BACAREX is a heavyweight ontology of planning objects and activities related to the forest fighting plan in the Andalusian regional

government. For every object stored, the ontology records both operational (e.g. geographic coordinates of the object) and informational metadata (i.e. information that may be needed by the technical staff during a forest fire incident, e.g. the radio channel of the responder responsible for a specific forest sector).

Conclusively, the existing third-party ontologies reported above share the drawback of covering only a subset of the notions involved in climate-related crisis management. Contrary to those, the beAWARE ontology semantically represents all aspects pertinent to crisis management, as indicated in Table 1, and as further described in the next chapter.

1.3 Ontology Engineering

For the design and formalization of the beAWARE ontology we considered the most recent established **ontology engineering methodologies**: METHONTOLOGY [Fernandez et al., 1997], DILIGENT [Pinto et al., 2004], On-To-Knowledge [Sure et al., 2004], DOGMA [Jarrar and Meersman, 2008], and NeOn [Suárez-Figueroa et al., 2009]. Other less modern approaches include Cyc [Lenat and Guha, 1989], Unified [Uschold, 1996], KACTUS [Bernaras et al., 1996], Sensus [Swartout et al., 1997], and the approach proposed by Grüninger and Fox [Grüninger and Fox, 1995].

From the above list of ontology engineering approaches, **we selected the NeOn methodology** for developing the beAWARE ontology model. Overall, NeOn is a modern scenario-based methodology that guides the ontology engineer to thoroughly define the requirements and characteristics of the ontology, supporting the reuse/reengineering of existing knowledge resources. The methodology's comparative advantages are:

- **Extremely well documented**, providing detailed guidance for all key aspects of the ontology engineering process.
- **Highly adaptable to project requirements** (people involved, end-users, domain(s) of interest, etc.).
- **Covers a multitude of ontology development scenarios**, capitalizing on existing ontologies and other relevant resources.
- **Structured representation of requirements** that formally describes the development requirements.
- **Exceptionally suited for collaborative ontology development**, as is also the case in beAWARE, with the ontology being collaboratively developed by project partners CERTH and IOSB, with frequent feedback by UPF.
- **Facilitates the adoption of Ontology Design Patterns (ODPs)** [Gangemi and Presutti, 2009] in the development process, increasing the standardization level and reinforcing the use of best practices and reusable successful solutions.

The steps we followed according to the adopted ontology engineering approach for designing and implementing the beAWARE ontology v1 are described in the next chapter.

1.4 Chapter Summary

This chapter presented all the relevant notions underlying the work presented in this deliverable. First, we introduced the concept of ontologies along with the pivotal role they play in the Semantic Web towards providing the foundations for machine-interpretable semantic knowledge representation and exchange. Then, we presented the most prominent existing ontologies for crisis management, and discussed the comparative advantages that the proposed beAWARE ontology brings into play. Finally, we presented the most recent established ontology engineering methodologies and justified our choice of the NeOn methodology for designing and developing the beAWARE ontology model. We have now set the stage for the next chapter, which will present in more detail the first iteration of the beAWARE knowledge base.

2 The beAWARE KB v1

This chapter discusses the scope, intended uses and requirements of the **beAWARE knowledge base (KB)** (also referred to as **ontology**), and presents its first iteration. The beAWARE ontology will be further refined during the pilot use case trials, and its second and final iteration will be presented in the upcoming deliverable D4.3, which is due M34.

2.1 Specification of Ontology Requirements

A key step in designing the ontology is to come up with a set of **Competency Questions (CQs)**, i.e. queries expressed in natural language that express a pattern for a type of question the knowledge base should be able to answer [Grüninger & Fox, 1995]. The answerability of CQs, hence, becomes a functional requirement of the ontology.

After carefully going through the project's user requirements reported in beAWARE deliverable D2.1 [Norbiato et al., 2017], and through close collaboration with the end user partners, we came up with a list of CQs that the ontology should be able to respond to, such as providing the location of a specific media item (e.g. a tweet, video, image etc.), or indicating the number and type of vulnerable objects detected from videos.

Note that the beAWARE ontology semantically represents three core aspects: (a) **climate-related natural disasters**, (b) **analysed data coming from the multimodal sensors**, and, (c) **rescue team assignments**. Thus, Table 2 includes the CQs per aspect that served as the foundation for the design of the ontology.

Table 2. Competency Questions (CQs) that drove the design of the beAWARE ontology v1.

1. Representation of natural disasters and relevant climate parameters, incidents and impacts.	
CQ1-1	<i>Which natural disasters may lead to natural disaster [X]?</i>
CQ1-2	<i>What are the impacts caused by natural disaster [X]?</i>
CQ1-3	<i>Which climate parameters characterize natural disaster [X]?</i>
CQ1-4	<i>What are the measurements for climate parameter [X] for natural disaster [Y]?</i>
CQ1-5	<i>What is the [maximum/minimum/average/...] measurement for climate parameter [X] during natural disaster [Y]?</i>
CQ1-6	<i>Where did natural disaster [X] take place?</i>
CQ1-7	<i>What incidents are associated with natural disaster [X]?</i>
CQ1-8	<i>Where did incident [X], which is associated with natural disaster [Y], take place?</i>
CQ1-9	<i>What are the impacts caused by incident [X] during natural disaster [Y]?</i>
CQ1-10	<i>What is the location with the [most/least] incidents during natural disaster [X]?</i>
CQ1-11	<i>What incidents took place during time interval [t₁] to [t₂] during natural disaster [X]?</i>
CQ1-12	<i>Which incidents during natural disaster [X] are the [most/least] severe?</i>
CQ1-13	<i>What is the priority of incident [X] during natural disaster [Y]?</i>
CQ1-14	<i>What incidents during natural disaster [X] are the most urgent (i.e. with the highest priority)?</i>
2. Representation of analysed data from the multimodal sensors.	
CQ2-1	<i>When and where was media item [X] created?</i>

CQ2-2	<i>What is the location with the [most/least] media items?</i>
CQ2-3	<i>Which vulnerable objects were involved in incident [X]?</i>
CQ2-4	<i>What impact do the vulnerable objects involved in incident [X] suffer?</i>
CQ2-5	<i>What is the risk suffered by vulnerable objects involved in incident [X]?</i>
CQ2-6	<i>What are the vulnerable objects that suffer the [greatest/smallest] risk during incident [X]?</i>
CQ2-7	<i>What is the detection confidence level for vulnerable object [X] during incident [Y]?</i>
CQ2-8	<i>What are the vulnerable objects with the [highest/lowest] confidence level detected during incident [X]?</i>
CQ2-9	<i>Which media items led to the creation of incident [X]?</i>
3. Representation of rescue unit assignments.	
CQ3-1	<i>What is the location of rescue unit [X]?</i>
CQ3-2	<i>What is the mission assigned to rescue unit [X] and what is its current status?</i>
CQ3-3	<i>What is the location where rescue mission [X] is taking place?</i>
CQ3-4	<i>What is the incident that rescue unit [X] is addressing?</i>
CQ3-5	<i>What are the vulnerable objects involved in the mission assigned to rescue unit [X]?</i>
CQ3-6	<i>What is the potential impact of the mission assigned to rescue unit [X]?</i>
CQ3-7	<i>What rescue missions have taken place during time interval $[t_1]$ to $[t_2]$?</i>
CQ3-8	<i>Where is the most urgent mission (i.e. the one with the highest priority) taking place?</i>
CQ3-9	<i>Which rescue unit is assigned the most severe incident?</i>

2.2 Reuse of Existing Resources

A common practice in ontology engineering involves the **reuse of existing third-party models**; this way we rely on previously used and validated ontologies for developing (parts of) our beAWARE ontology.

Starting with the representation of environmental and meteorological conditions, we rely to some extent on the PESCaDO ontologies [Rospocher & Serafini, 2012], and, more specifically, we adopted a number of related properties from classes **EnvironmentalData** and **EnvironmentalNode**.

Regarding the representation of disaster impacts, we relied on MOAC [Limbu, 2012], mainly classes **AffectedPopulation**, **CollapsedStructure**, **CompromisedBridge**, **Deaths**, **InfrastructureDamage** and properties **affectedby** and **impact**. Moreover, for categorizing damages and resources, we are based on SoKNOS [Babitski et al., 2011], and, more specifically on the SoKNOS approach for representing damages and their association to resources [Babitski et al., 2009].

Furthermore, with regards to rescue unit assignments, our adopted representation is based on the approach proposed by the OASIS project [Couturier & Wilkinson, 2005], mainly the part for representing mission assignments to units and associating missions to incidents taking place during a crisis.

Finally, the beAWARE ontology v1 imports the Simple Knowledge Organization System (SKOS) [Miles & Bechhofer, 2009], which provides a set of metadata fields for enriching the

ontology documentation. Specifically, we used **skos:definition** for providing the definitions of the classes and properties, and **skos:example** for providing examples of usage.

2.3 Ontology Conceptualization

This subsection describes the conceptualization of the ontology, in order to satisfy the ontology requirements represented above as CQs. The models presented here are based on the **Grafoo ontology visualization notation** [Falco et al., 2014]. Figure 2 displays an overview of the core ontology classes; for simplicity, we have omitted data type and inverse properties, as well as extensive class hierarchies. The full list of classes and properties is presented in Appendix A.

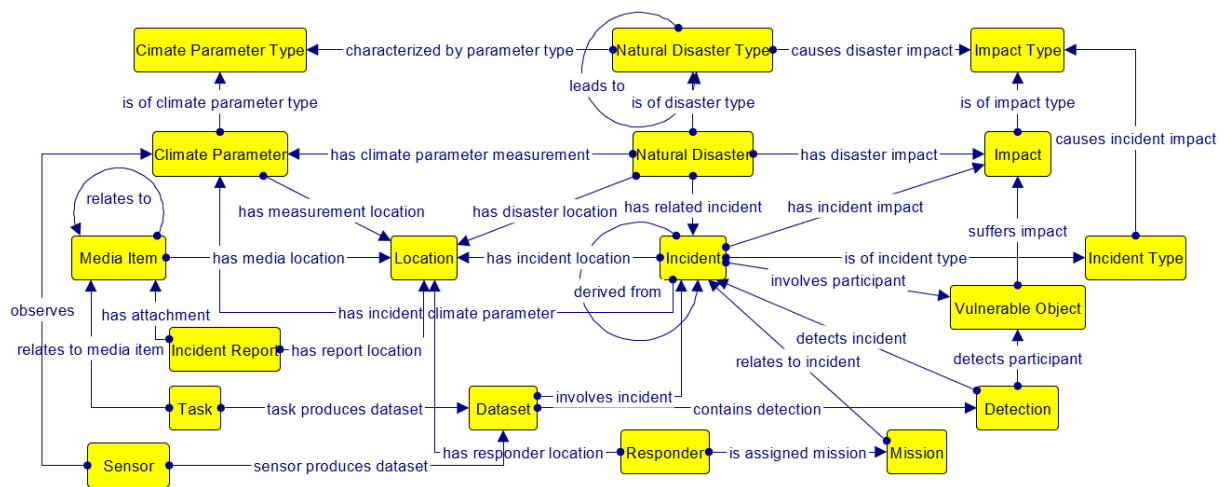
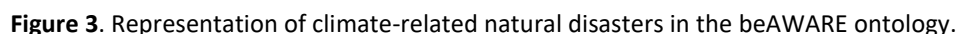


Figure 2. High-level overview of the core classes of the beAWARE ontology.

The following subsections present the three core aspects of the ontology, as they were introduced before: (a) representation of natural disasters, (b) representation of analysed data coming from the multimodal sensors, and, (c) representation of rescue team assignments.

2.3.1 Representing Natural Disasters

The representation of climate-related natural disasters in beAWARE ontology v1 is illustrated in Figure 3.



Note that several notions in the ontology come in pairs, **<Notion>** and **<Notion Type>**, associated via property **<is of type>**. This approach allows us to integrate two distinct layers of expressiveness into the ontology: **<Notion Type>** is the more abstract layer for interconnecting notions at a higher level (e.g. what types of impacts are caused by hurricanes? what are the climate parameters that characterize a heatwave?). While **<Notion>**, on the other hand, allows us to represent the actual manifestations of the notions and contains all metadata for the specific event. For instance, the **UK heatwave** in Figure 4 is a manifestation of the **Heatwave** natural disaster type. This dual scheme is adopted in the whole beAWARE ontology.



¹ <http://www.bbc.com/news/uk-40353118>

2.3.2 Representing Analyzed Data

Besides the representation of climate-related natural disasters and pertinent notions, the beAWARE ontology also encompasses information relevant to the analysis of input data coming from the various sensors of the framework. This information is fed to the ontology from the analysis components; the core constructs in the ontology are illustrated in Figure 5.

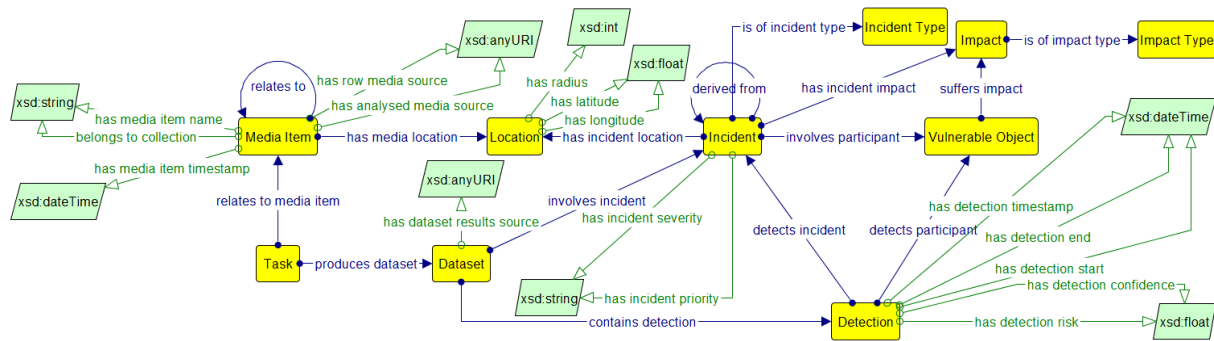


Figure 5. Representation of analyzed data in the beAWARE ontology.

Class **Media Item** represents an item of analyzed data, which is related to some analysis task (via class **Task**). Media items can be pieces of text, images, videos, or social media posts, all of them submitted during the occurrence of the crisis. The analysis of the respective items (text analysis, image analysis or video analysis) produces a **Dataset** containing all relevant information (e.g., an object detection task may produce a dataset of detected incidents, objects, and confidence scores).

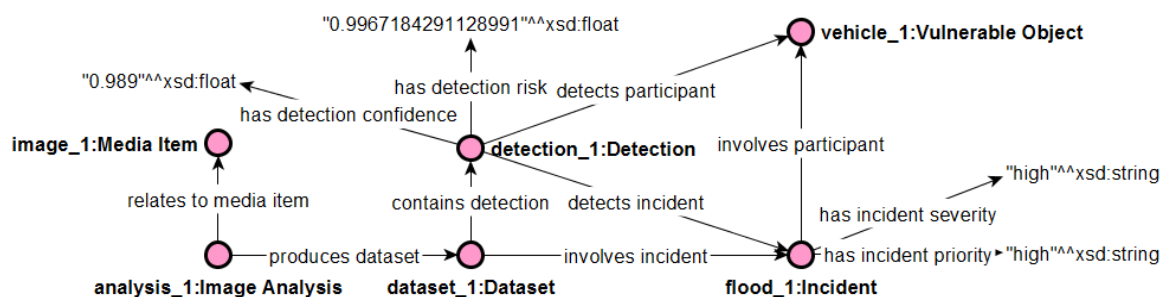


Figure 6. A video analysis example in the beAWARE ontology.

Figure 6 demonstrates an example of a video analysis instance, where a vehicle is detected participating in a flood incident. Note that the beAWARE ontology contains a complete typology of media items (text, image, video, social media), vulnerable objects (e.g. assets, stakeholders, infrastructure, buildings etc.), impacts, data analyses, and incidents.

2.3.3 Representing Rescue Team Assignments

The third component of the beAWARE ontology is responsible for semantically representing rescue team assignments. This component is not very mature yet, but will be extended in the next iteration of the ontology.

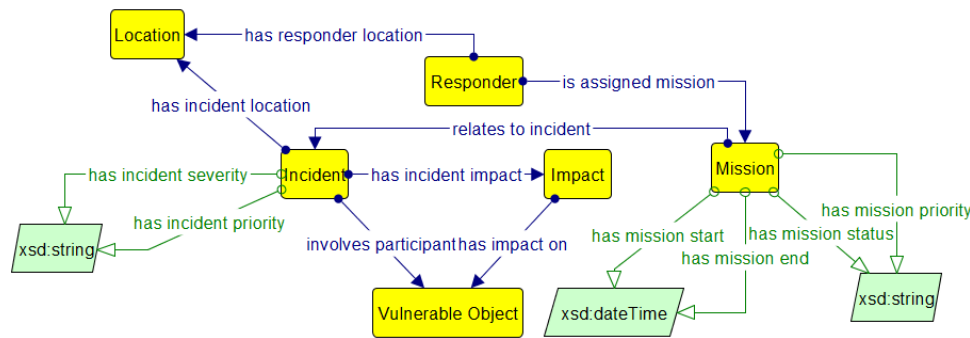


Figure 7. Representation of mission assignments to first responder units in the beAWARE ontology.

Figure 7 displays the respective concepts in the proposed ontology. First responders (class **Responder**) are assigned one or more missions (class **Mission**), which in turn relate to incidents that involve participating entities (class **Vulnerable Object**). A mission is also characterized by start and end time, status and mission priority.

2.4 Semantic SensorThings Mapping

In order to represent information related to a sensor and its readings, we are relying on the data model described in the **SensorThings API standard** [Liang, 2016; OGC_1 2017]. The OGC SensorThings API standard defines a data model and a REST-API to access the data. It can be described as Sensor Web Enablement for the Internet of Things. It is a modern, light-weight REST API designed for storing and requesting sensor data, with advanced filtering options. The data model of this standard is based on the OGC/ISO Observations and Measurements model [OGC_2, 2017].

2.4.1 Base Model

The data model of the OGC SensorThings API consists of eight entities, with their properties and relations (see Figure 8). The entities are:

- **Thing**: A virtual or physical object. Depending on the use case, this entity can be the object being observed, or the sensor platform, such as a weather station or an unmanned aerial vehicle.
- **Location**: The locations of Things. These can be geographic locations, encoded as points or areas, or symbolic locations, like “Mateotti Square”.
- **HistoricalLocation**: the link between a Thing and a Location, with the time indicating when the Thing was in a certain Location.
- **Sensor**: A sensor that can generate data.
- **ObservedProperty**: A property of the Feature of Interest that is being observed by a sensor. For instance, the temperature in a city, or the river level in a river section.
- **Datastream**: a collection of Observations of one ObservedProperty, made by one Sensor, and linked to one Thing.

- **Observation**: a measurement made by a Sensor.
- **FeatureOfInterest**: The geographic area or location for which an Observation was made. This can be the same as the Location of the Thing, which is often the case for in-situ sensing. In the case of remote sensing, the feature of interest can be different from the location of the Thing, depending on what is chosen as the Thing. The feature is a geographical point or a polygon encompassing an area or volume, usually encoded in GeoJSON.

The relations between these entities are also defined by the data model (see Figure 8). Most relations are one-to-many: An **Observation** must have one **FeatureOfInterest** and one **Datastream**, while a **Datastream** and **FeatureOfInterest** can have zero or more **Observations**. A **Datastream** must have one **ObservedProperty**, one **Sensor** and one **Thing**, while a **Thing**, **ObservedProperty** and **Sensor** can have zero or more **Datastreams**. A **HistoricalLocation** must have one **Thing**, while a **Thing** can have zero or more **HistoricalLocations**.

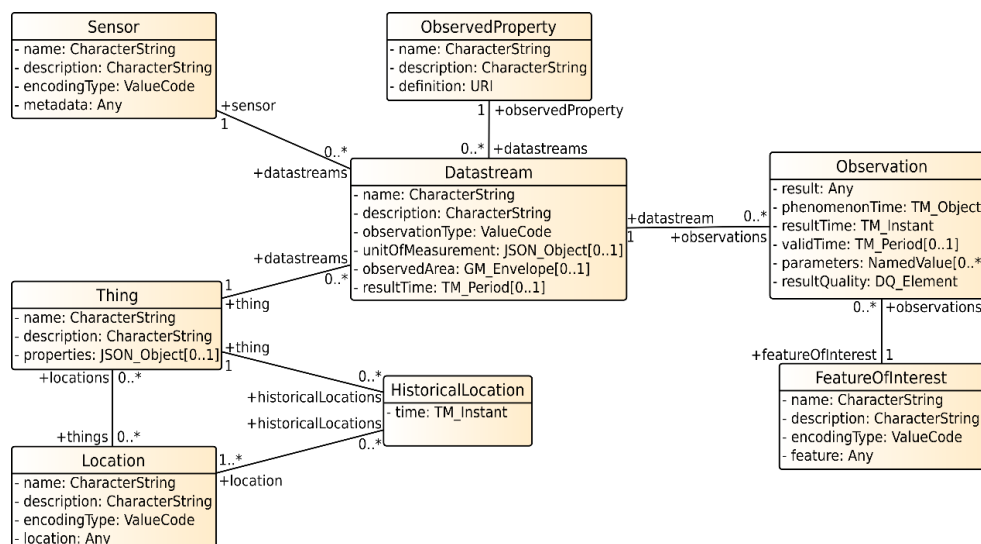


Figure 8. The OGC SensorThings API data model.

Multiple relations exist for **Location**: A **Thing** can have zero or more **Locations**, but these **Locations** must all be different representations of the same physical location; for instance, one geospatial location, represented by GPS coordinates, and one symbolic location. A **Location** can have zero or more **Things**.

Each time a **Thing** is linked to a new **Location** (or set of **Locations**), a new **HistoricalLocation** is generated that tracks the time when the **Thing** was at this **Location**. A **HistoricalLocation** also has the restriction that if it has more than one **Location**, these **Locations** have to be different representations of the same real-world location.

When applied to beAWARE, for example to a water level sensor in a specific river section, the Thing could be the river section within which the sensor is measuring the water level.

The **Thing** would have a **Location** with the position of the river section. Since the section cannot move, there would be only one **HistoricalLocation**. The **Sensor** entity would describe the exact properties of the sensor, like type and brand. The **ObservedProperty** would be named “water level” and contain an exact reference to the water level entry in the knowledge base. For this set of **Thing**, **Sensor** and **ObservedProperty** there would be a **Datastream**, grouping the observations for this sensor in this section. Each value measured by the sensor would be stored as an **Observation**. Since the sensor is static, each **Observation** is linked to the same **FeatureOfInterest**, which has the position of the river section.

For a water velocity sensor in the same system, the same **Thing**, **Location** and **FeatureOfInterest** entities would be used. Only new **Sensor**, **ObservedProperty** and **Datastream** entities would need to be added.

2.4.2 MultiDatastream

In some cases, a sensor produces multiple, related, result values. For instance, a weather station measuring wind usually returns a value for wind speed and a separate value for wind direction. A user requesting these kinds of sensor data will almost always want all of these related values grouped together.

To support this type of observation with multiple result values, the SensorThings API has the **MultiDatastream** entity. A **MultiDatastream** is similar to a **Datastream**, but instead of only one **ObservedProperty** and one unit, a **MultiDatastream** has multiple **ObservedProperties** and units. The **Observations** in a **MultiDatastream** have a JSON array as result, containing the same number of values as the **MultiDatastream** has **ObservedProperties**.

2.4.3 Actuation

The OGC specification “OGC SensorThings API Part II - Tasking Core”, released for public comment on the 20th of February, 2018, adds actuation to the SensorThings API. Using this standard it is possible to describe the parameters used to control actuators, and to define tasks that an actuator should execute. The term actuator here is not limited to hardware devices; an actuator can be anything that can accept a task to be executed, including mathematical models.

The Tasking Core adds three entities to the SensorThings API: **Actuator**, **TaskingCapability** and **Task**:

- **Actuator**: A hardware or software component that can execute tasks.
- **TaskingCapability**: Describes the parameters that can be supplied to a task.
- **Task**: A task to be executed, with values for the parameters described in the linked **TaskingCapability**.

2.4.4 Mapping the Sensor Model to the beAWARE Ontology

The representation of sensor data and metadata in the ontology requires that the entities of the SensorThings API are mapped to concepts in the ontology and linked to the other relevant concepts. The result of this mapping is that each entity in the data model of the SensorThings API is mapped in a concept of the beAWARE ontology. This mapping process is ongoing and will be finalized in the final iteration of the ontology; the current mapping is presented in Table 3.

Table 3. Mapping of the SensorThings API entities to the beAWARE ontology concepts.

SensorThingsAPI Entity	beAWARE Ontology Concept
Sensor	Sensor
Feature of Interest	<not yet implemented>
Location	Location
Observation	Climate Parameter
Datastream	Dataset
Thing	<not yet implemented>
Observed Property	Climate Parameter Type
Historical Location	<not yet implemented>

Note that the SensorThings API entity **ObservedProperty** is mapped to concept **Climate Parameter Type** in the ontology, but will need to be extended to other types of non-climate observations. A **Datastream** is mapped to **DataSet**. Furthermore, since entities in the SensorThings API can be referenced directly with a URL, the ontology can contain those references, allowing for easy navigation between the ontology and the data in a SensorThings API service. Since the service is REST-based, it is also possible to include a search query in the URL.

2.5 Ontology Formalization and Implementation

The beAWARE ontology expressed in **OWL 2** [W3C, 2012], a knowledge representation language widely used within the Semantic Web community for developing ontologies. Thus, we capitalize on its wide adoption as well as its formal structure and syntax, which is based on **Description Logics (DLs)**, a family of knowledge representation formalisms characterised by logically grounded semantics and well-defined reasoning services.

The main building blocks of DLs 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 $\exists \text{hasFriend}.\text{Person}$ 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.

For developing and deploying the ontology we relied on the following tools:

- **Protégé-OWL v5.0** [Musen, 2015], the popular ontology development environment;
- **GraphDB**² for locally hosting test versions of the ontology;
- **SPARQL** [Harris et al., 2013] served as the semantic query language for submitting queries to the ontology and running rules on top of the model;
- **YASGUI**³ for formalizing the SPARQL queries.

Note that the official repository for hosting the ontology is **WebGenesis** (see next chapter), which is being developed and maintained by project partner IOSB.

A publicly available version of the beAWARE v1 ontology can be found at the following URL: <https://goo.gl/5VR4qB>

2.6 Ontology Evaluation

This subsection presents an evaluation of the beAWARE ontology with regards to its consistency, quality, structure and compliance with user requirements.

2.6.1 Evaluating the Consistency and Quality

For evaluating the overall consistency and quality of the ontology we used **OOPS (Ontology Pitfall Scanner)**, an online tool for detecting the most common pitfalls⁴ in ontologies [Poveda-Villalón et al., 2014]. After analyzing the ontology, OOPS provides a list with all the pitfalls it detected along with the associated negative consequences, and suggests modifications in order to improve the quality of the ontology. The tool can detect:

- **Critical pitfalls** affecting the ontology's consistency, which are crucial to be corrected;
- **Important pitfalls**, which are not equally critical, but are considered also important to be corrected;
- **Minor pitfalls**, which do not cause any critical problems, but correcting them will improve the quality of the ontology.

We submitted a prior version of the ontology to OOPS for the purposes of the work presented in [Kontopoulos et al., 2018] and corrected all the detected pitfalls. The current version of the ontology has no more pitfalls, with the exception of some pitfalls concerning the imported SKOS ontology (see subsection 2.2), which were, thus, left unresolved.

² <https://ontotext.com/products/graphdb/>

³ <http://yasgui.org/>

⁴ A catalogue of common pitfalls is given at <http://oops.linkeddata.es/catalogue.jsp>

2.6.2 Evaluating the Structure

For evaluating the structure, we relied on **OntoMetrics**⁵, an online framework that validates ontologies based on established metrics. Table 4 presents the results derived from the analysis by OntoMetrics. Base Metrics comprise of simple metrics, like the count of classes, axioms, objects etc.; these metrics show the quantity of ontology elements. Schema metrics, on the other hand, address the design of the ontology; metrics in this category indicate the richness, width, depth, and inheritance of an ontology schema design.

Table 4. Ontology metrics for the beAWARE ontology v1, generated by OntoMetrics.

Base Metrics	Class count	68
	Object property count	75
	Data property count	25
	SubClassOf axioms count	46
	Disjoint classes axioms count	6
	Inverse object properties axioms count	31
	Functional object property axioms count	1
	Transitive object property axioms count	6
	Symmetric object property axioms count	5
	DL expressivity	$SI^{(D)}$
Schema Metrics	Attribute richness	0.367647
	Inheritance richness	0.676471
	Relationship richness	0.637795
	Axiom/class ratio	11.161765
	Inverse relations ratio	0.413333
	Class/relation ratio	0.535433

Starting with the base metrics, the total count of classes and properties indicates that the proposed ontology is a rather lightweight model, which could be easily adopted by various applications, contrary to heavier “monolithic” ontologies that pose significant challenges in integration. Furthermore, DL expressivity refers to the Description Logics variant the ontology belongs to (see also subsection 2.5). $SI^{(D)}$ indicates a simple ontology (universal restrictions, limited existential quantification) with inverse, transitive, and datatype properties.

Regarding schema metrics, the measurements in the table are adopted from [Gangemi et al., 2005] and [Tartir et al., 2010]. **Attribute richness** is defined as the average number of attributes per class and can indicate both the quality of ontology design and the amount of information pertaining to instance data. The more attributes that are defined the more knowledge the ontology conveys. **Inheritance richness** is defined as the average number of

⁵ <https://ontometrics.informatik.uni-rostock.de>

subclasses per class and is a good indicator of how well knowledge is grouped into different categories and subcategories in the ontology. This measure can distinguish a horizontal ontology (where classes have a large number of direct subclasses) from a vertical ontology (where classes have a small number of direct subclasses). The respective value in the table indicates that the proposed ontology is somewhere in between; this is reasonable, since the ontology covers many aspects (breadth) while thoroughly modelling some of them (depth). **Relationship richness** refers to the ratio of the number of non-inheritance relationships (i.e. object properties, equivalent classes, disjoint classes) divided by the total number of inheritance (i.e. subclass relations) and non-inheritance relationships defined in the ontology. This metric reflects the diversity of the types of relations in the ontology. Finally, **axiom/class ratio**, **class/relation ratio**, and **inverse relations ratio** describe the ratio between axioms-classes, classes-relations, and inverse relations-relations, respectively, and are indications of the ontology's transparency and understandability.

2.6.3 Compliance with User Requirements

As discussed in section 2.1 , user requirements are mapped to CQs that the ontology is expected to answer. Following the methodology proposed in [Zemmouchi-Ghomari & Ghomari, 2013], we translated the CQs into SPARQL queries and evaluated the retrieved results. Appendix B includes the set of CQs, along with their SPARQL translation and an evaluation of the retrieved result sets. As seen in the table, all of the CQs have been evaluated positively.

2.7 Chapter Summary

This chapter presented the first iteration of the beAWARE ontology. Starting with the user requirements, we formulated the competency questions underlying the ontology design and then presented the set of existing third-party resources we used for implementing parts of the knowledge base. We then presented the conceptualization of the three core aspects of the ontology, and we also discussed the mapping of the beAWARE ontology to the SensorThings model, for representing input coming from the various sensors. Finally, we presented the evaluation of the ontology's consistency and quality, as well as the evaluation of its structure and the verification of its compliance to the user requirements. The following chapter will introduce WebGenesis, the repository for hosting the ontology.

3 Knowledge Base Repository

WebGenesis is the platform that hosts the beAWARE KB Repository (KBR). Developed by beAWARE partner IOSB, WebGenesis is a content, community, and knowledge management system, providing several options for manipulating, requesting, inserting and visualizing data. Unlike many other Content Management Systems (CMSs), it is also able to work with ontologies. This chapter outlines the capabilities of the platform regarding ontology management and maintenance.

3.1 WebGenesis Characteristics and Scalability

WebGenesis separates content and layout, offering standardized templates for information categories and harmonizes the appearance. If the standard formatting does not meet the user's needs, one can customize every entry with HTML.

With regards to scalability, the system is highly dependant on the environment it is deployed in. For example: in another project several thousand users had access to the platform at the same time. During this stress test, WebGenesis provided reliable support and functionality.

Moreover, WebGenesis supports several databases and can also use distributed or clustered databases, if they are accessible via a single JDBC interface. Since it is programmed in Java, the performance can easily be improved by allocating RAM to the Java Virtual Machine.

3.2 Navigating the beAWARE Knowledge Base

The default layout of WebGenesis has been customised for the beAWARE project. Figure 9 shows the entry point to the ontology within WebGenesis.

The tree diagram on the left shows the concepts within the ontology, and one can navigate the ontology by clicking an entry in this tree. Alternatively, it is possible to navigate through the ontology diagram on the right, by clicking on one of the elements inside. These diagrams have been created to help the user to better understand the beAWARE ontology, its relations and the network of knowledge, and can be customized to show the user-required relations. This feature will be discussed in subsection 3.3 .

Clicking a concept in the tree gives a detailed view of a class (see Figure 10). All object properties of a class can be seen at once under *Object Properties*. The subject, predicate and object for each triple can be selected to learn more about them. Similarly, all datatype properties describing the class can be found in *Datatype Properties*, while, in order to view the instances of a specific class, one should choose *Show Instances* from the *Actions* tab. This will bring a list of all instances of this class and a preview of their description.

Clicking on an instance will deliver all information about this instance (see Figure 11). The left side gives information, whereas the right side shows the relationships with other

instances. The right side is clickable, enabling the user to get a deeper understanding and find out about causalities or the properties of related instances. The right down side also offers the possibility to access documents of eventual interest.

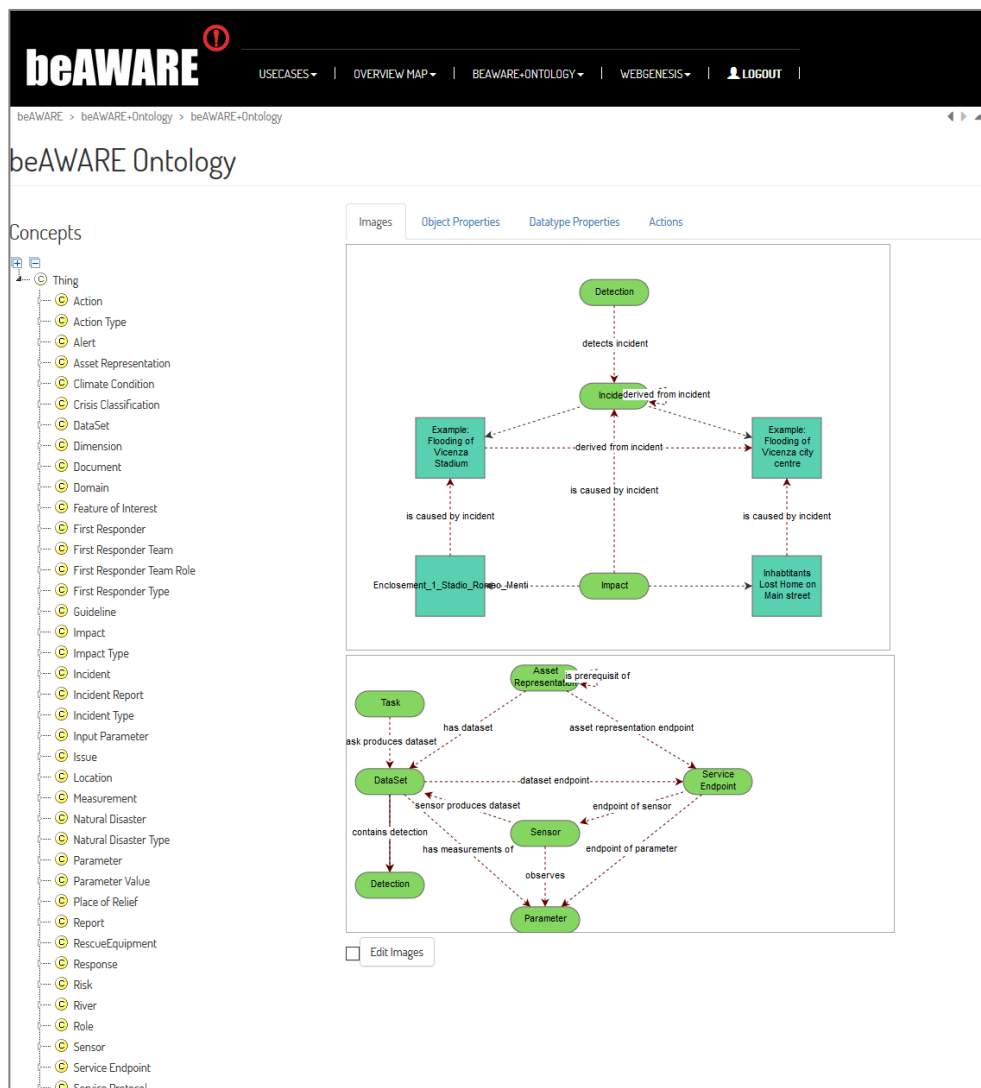


Figure 9. The entry point of the beAWARE ontology in WebGenesis.

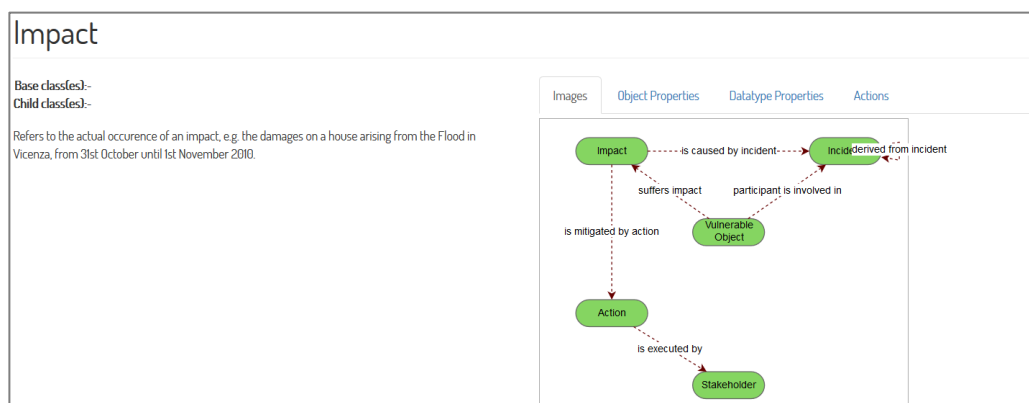


Figure 10. Displaying a class in WebGenesis.



The screenshot displays the beAWARE WebGenesis interface. The top navigation bar includes the beAWARE logo, a notification icon, and links for USECASES, OVERVIEW MAP, BEAWARE-ONTOLOGY, WEBGENESIS, and LOGOUT. The breadcrumb trail indicates the current view: beAWARE > beAWARE-Ontology > Instances > IncidentIndividuals > Example: Flooding.

Example: Flooding of Vicenza city centre

The picture shows the flooding of the Vicenza city center with flooded palace and streets.



Unique name: Example: Flooding
Text: The picture shows the flooding of the Vicenza city center with flooded palace and streets.
Display name: Example: Flooding of Vicenza city centre

Edit Images

- Concept**
 - Incident
- Derived from incident**
 - Example: Rain In Bacchiglione Basin Area
- Has incident location**
 - Location Main Street Vicenza
- Is of incident type**
 - Flood
- Lead to incident**
 - Example: Flood in South Vicenza
 - Example: Flooding of Vicenza Stadium
- Is dataset detection of**
 - Data Set Bacchiglione Basin Water Level
 - Data Set From Video Analysis
 - Detection Video Analysis City Center
 - Video Analysis_detection_01
 - Video Analysis_detection_02
 - Video Analysis_detection_03
 - Video Analysis_detection_04
- Incident is detected by**
 - Detection Video Analysis City Center
 - Video Analysis_detection_01
 - Video Analysis_detection_02
 - Video Analysis_detection_03
 - Video Analysis_detection_04

Figure 11. Viewing an instance in WebGenesis.

3.3 Editing an Ontology Image

On the right side of the main ontology entry and each class, the user can create one or more diagrams for the class relationships (see Figure 12). By clicking on *Edit Images* the edit mode will be enabled. Choosing *Add New Image* creates a new image only containing the class that is displayed at the moment, while other classes and instances can be added as required: right clicking an element shows all available super classes, subclasses, object properties or instances that are related to or contained in the concept. Classes are shown in green roundish figures. Instances are shown in angled turquoise figures. A continuous arrow indicates parent-child class relations. Dotted arrows indicate object properties.

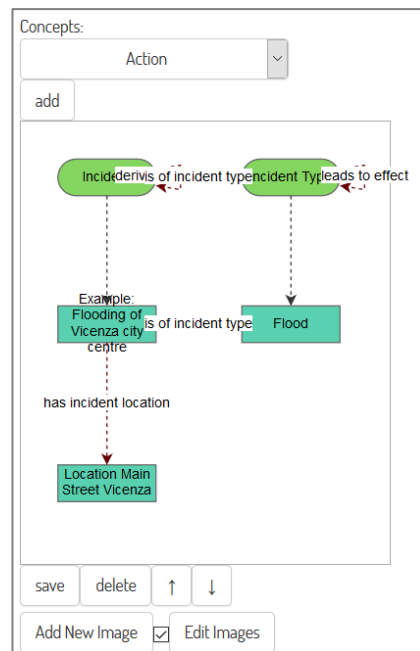


Figure 12. Editing ontology images in WebGenesis.

3.4 Populating the Ontology

To populate the knowledge base, two possible approaches can be distinguished. First, WebGenesis offers the insertion of information via an Input Form. These forms can be created for any class from the *Action Tab – Create input form*, and WebGenesis will display the form in Figure 13.

Create input form for Incident

DatatypeProperty for title

display name

DatatypeProperty for name

unique name

DatatypeProperty for text

Order

- http://beaware-project.eu/beAWARE/#instanceDisplayName
- http://beaware-project.eu/beAWARE/#instanceUniqueName
- http://beaware-project.eu/beAWARE/#hasIncidentPriority
- http://beaware-project.eu/beAWARE/#hasIncidentSeverity
- http://beaware-project.eu/beAWARE/#instanceText
- http://beaware-project.eu/beAWARE/#derivedFormIncident
- http://beaware-project.eu/beAWARE/#isOffIncidentType
- http://beaware-project.eu/beAWARE/#leadToIncident
- http://beaware-project.eu/beAWARE/#isDatasetDetectionOf
- http://beaware-project.eu/beAWARE/#incidentIsDetectedBy
- http://beaware-project.eu/beAWARE/#hasIncidentImpact
- http://beaware-project.eu/beAWARE/#involvesParticipant
- http://beaware-project.eu/beAWARE/#hasIncidentLocation
- http://beaware-project.eu/beAWARE/#EndingTime

Reference Properties by

☒ URI of the Property

☐ ID of the entry

☐ Create file upload field

Cancel Create

Figure 13. Creating an input form in WebGenesis.

Here the user can link title and name for the new instances to elements of the input form that is being created. The box order lets you change the order of appearance of the text boxes and forms in the input form. Clicking create will store the input form as *Inputform for [Class]* and save it in a respective folder *Input Forms*, which is located in the same folder as the ontology entry.

After clicking on any input form, one can insert the name and all information of the instance that shall be stored into the KB. If the new instance is interlinked with other information in the knowledge base, the referring instances can be selected by clicking the *Select* button in the form. The user can choose the desired instances here – no instance is also a valid option. After confirming and saving the instance with *Save*, the user is redirected to the newly created instance. Note that, while this is not the intended use in beAWARE (where the ontology will be populated by analysis algorithms), it is a helpful for creating test scenarios and sample data.

An alternative ontology population mode is through the WebGenesis REST interface. As seen below, the header must indicate the Content-Type as *multipart/form-data* and Accept as *application/json*. This is possible via a POST request to the following address, described in the box below. The body contains the user credentials.

```
https://beaware.server.de/servlet/is/rest/grantee/login/

<Header>

Content-Type:application/x-www-form-urlencoded
Accept:application/json

<Body>

aspect:doLogin
user:YourUserName
key:YourPassword
```

Also, one can authenticate the client with a GET request containing the user credentials, as seen below:

```
https://beaware.server.de/servlet/is/rest/grantee/login/?user=YourUs
ername&key=YourPassword
```

Now, the interface can be accessed by sending JSON via POST request to the interface URLs written below.

- <https://beaware.server.de/servlet/is/rest/entry/1932/addABoxData/>
- <https://beaware.server.de/servlet/is/rest/entry/1932/removeABoxData/>

After authentication, a REST client can post data to the endpoint. An example is shown below. In the header, application and content-type must be set to *application/json*. The client has to provide the new instance, its relationships to other instances and eventual alphanumerical values in *data*. After posting, the KB publishes the new data. This is the

favourable approach to enable automated population of the knowledge base and is adopted by the KBS (see next chapter).

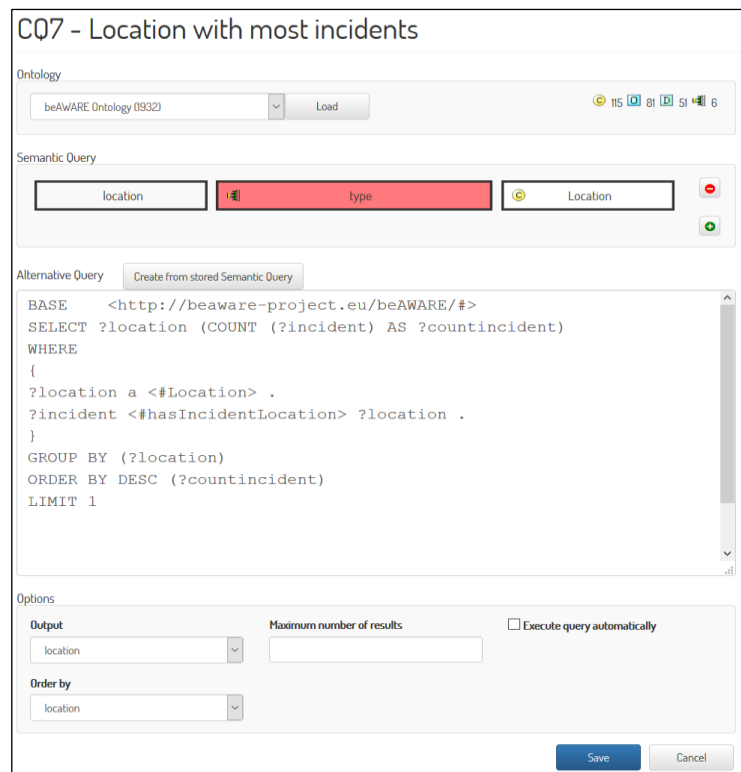
```
https://beaware.server.de/servlet/is/rest/entry/1932/addABoxData/  
  
<Header>  
  
application:application/json  
Content-Type:application/json  
  
<Body>  
  
{  
  "defaultprefix" : "http://beaware-project.eu/beAWARE/",  
  "data" : { ... }  
}
```

3.5 Requesting Data from the Ontology

The ontology has implicit and explicit attributes and interrelationships between classes – the instances are interlinked. Therefore, elements create a complex network, which can be queried. If specific data and their causalities are of interest, a SPARQL query will yield the desired result. One can configure a query with the user interface displayed in Figure 14. It supports the user by offering a graphical query editor, which helps creating necessary queries and translates them directly into SPARQL. If desired, the user can also manually enter SPARQL requests.

For SPARQL clients it is also possible to retrieve the information via an endpoint. To access the SPARQL endpoint, the client needs to identify itself. The process is similar to what was described in the previous subsection.

After authentication, a client can query data via HTTP POST or GET requests, while the response from WebGenesis will be delivered in JSON. It is also possible to include other knowledge bases in the query: WebGenesis allows the retrieval of data from remote SPARQL endpoints.



CQ7 - Location with most incidents

Ontology

beAWARE Ontology (932) Load

Semantic Query

location type Location

Alternative Query Create from stored Semantic Query

```
BASE <http://beaware-project.eu/beAWARE/#>
SELECT ?location (COUNT (?incident) AS ?countincident)
WHERE
{
  ?location a <#Location> .
  ?incident <#hasIncidentLocation> ?location .
}
GROUP BY (?location)
ORDER BY DESC (?countincident)
LIMIT 1
```

Options

Output location Maximum number of results Execute query automatically

Order by location

Save Cancel

Figure 14. The WebGenesis SPARQL-query editor.

For a POST request, the client must inform the server about the *Content-Type:application/sparql-query* in the header. The body contains the key *query* and the query itself (see below).

```
https://beaware.server.de/servlet/is/entry.1932.SPARQLEndpoint/

<Header>

Content-Type:application/sparql-query

<Body>

query:
BASE <http://beaware-project.eu/beAWARE/#>
SELECT ?incident (COUNT (?human) as ?cHuman)
WHERE
{
  ?impact <#isCausedByIncident> ?incident .
  ?human <#suffersImpact> ?impact .
  ?human a <#Human> .
}
GROUP BY (?incident)
ORDER BY ASC(?name)
```

A GET request sent to the endpoint contains the query in the URL. Therefore, the query must be URL encoded.

3.6 User Management

WebGenesis also offers user management, which allows the system to distinguish several users and limit their capabilities to what is necessary. The admin can grant specific access rights; the following access rights are available:

- No access
- Display
- Read
- Read and modify
- Add
- Add and modify
- Full Access without change access
- Full Access

3.7 Chapter Summary

This chapter presented WebGenesis, the system that hosts the beAWARE KB, and focused on its capabilities for ontology management and maintenance. The next chapter introduces another key component, the Knowledge Base Service, which plays the role of the interface with the ontology and acts as an intermediate layer between WebGenesis and the rest of the beAWARE modules.

4 Semantic Integration and Reasoning

As described in the previous chapter, the beAWARE KB is hosted on WebGenesis. On top of that, the **Knowledge Base Service (KBS)** is a system component that acts as an intermediate layer between the WebGenesis triplestore and the rest of the beAWARE modules. This chapter describes how system data are semantically integrated into the beAWARE ontology, allowing the application of semantic reasoning and the transfer of inferred, high-level knowledge back to other system components.

4.1 Semantic Integration

The KBS component is implemented in Python. It features subscribe/publish capabilities to the beAWARE communication bus (MSB), enabling a full-fledged two-way interaction with other system components for the exchange of information. Indicatively, a set of connections is presented in Figure 15.

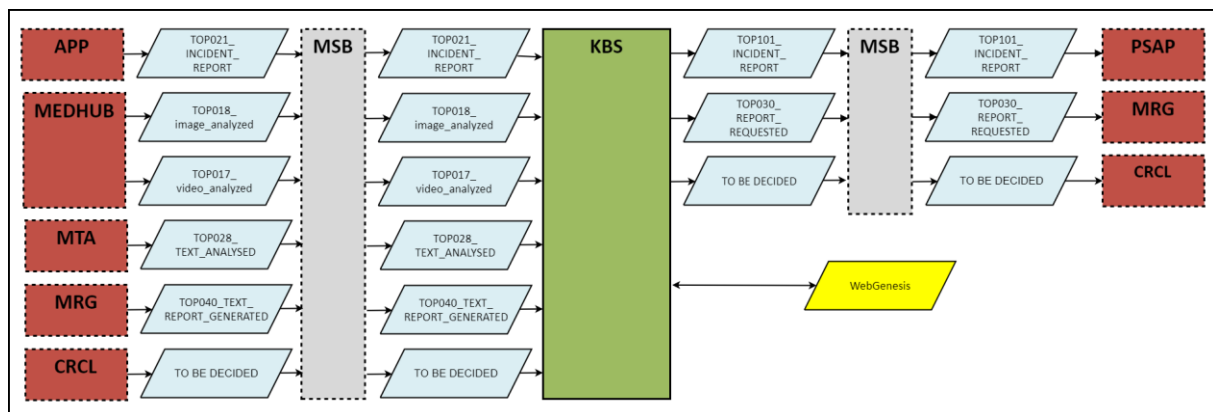


Figure 15. KBS connectivity diagram.

In a sense, the KBS monitors all reported information, sourced from civilians, first responders and sensors, along with analysis results from images, videos, audio and text. Consequently, these low- and high-level pieces of information are interrelated, mapped and semantically integrated (i.e. populated) into the appropriate beAWARE ontology concepts.

More specifically, the MSB allows the exchange of JSON-formatted messages within a set of predefined topics. These messages are parsed by the KBS and, based on the topic, specific parts of information from the message body are stored into the ontology. Then, the interested system components are informed via the MSB with an appropriately structured message. For instance, when a civilian posts an incident report using the beAWARE mobile application, a message with the following structure is broadcasted to the MSB:

```
{
  "header": {
    "topicName": "TOP021_INCIDENT_REPORT",
    "topicMajorVersion": 1,
```

```
{
  "topicMinorVersion": 0,
  "sender": "SCAPP",
  "msgIdentifier": "1578c96b-d0bc-4ba7-b65c-54e472fe94dc",
  "sentUTC": "2018-03-27T11:40:23Z",
  "status": "Actual",
  "actionType": "Alert",
  "specificSender": "mobileAppTechnicalUser",
  "scope": "Restricted",
  "district": "Thessaloniki",
  "recipients": "",
  "code": 0,
  "note": "",
  "references": ""
},
"body": {
  "incidentOriginator": "SCAPP",
  "incidentID": "incident_id_x",
  "language": "en-US",
  "startTimeUTC": "2018-03-27T11:40:23Z",
  "title": "Fire!",
  "position": {
    "latitude": 39.648914,
    "longitude": -0.300993
  }
}
}
```

Thereupon, the KBS, which is subscribed to topic *TOP021_INCIDENT_REPORT*, will identify this as an alert (*actionType*) of a new incident report, taking place at a certain position, at a specific time (*startTimeUTC*). This alert of an incident report also incorporates a unique ID (*incidentID*), which will help to match information from upcoming updates to this specific report. As a result, in accordance with the beAWARE ontology schema (see Chapter 2), new instances are populated into WebGenesis, into classes **IncidentReport** and **Location**. Additionally, the necessary connections between these instances are established, using object properties like **hasReportLocation**, while other knowledge is stored using datatype properties like **hasReportID**, **latitude** and **longitude**.

To describe this process in more detail, the KBS has two main Python classes that receive the incoming messages as arguments from the MSB consumer module and consequently handle them. The first class, called *Message2KB*, is responsible for identifying information in the messages and for populating them to the KB, while class *Reasoner* accesses the KB to apply semantic reasoning - after *Message2KB* has taken action - and forwards the inferred knowledge to the MSB, as described in the next subsection. Both classes have a similar constructor, where the message is formatted as a Python dictionary and the topic is extracted, as shown in the following code snippet:

```
self.message = json.loads(message_text)
self.topic = self.message['header']['topicName']
```

Then, the name of the topic (e.g. *TOP021_INCIDENT_REPORT*) acts as a guide to which path should be followed and what action should be taken. Therefore, for each different topic a homonymous class function has been defined (e.g. function *top021_incident_report()*) and is called with the following command:

```
# Run the corresponding topic function
try:
    getattr(self, self.topic.lower())()
except:
    pass
```

With regard to the earlier example of a new incident report, the *Message2KB.top021_incident_report()* function will first try to validate the existence of certain required fields:

```
try:
    incident_id = self.message['body']['incidentID']
    position_latitude =
self.message["body"]["position"]["latitude"]
    position_longitude=self.message["body"]["position"]["longitude"]
]

except Exception as e:
    print("Error @ Message2KB.top021_incident_report():\n", e)
    return
```

This is an important sanitization level to ensure that the KBS will not crush upon malformed messages on the MSB. Next, a JSON-formatted insert query will be assembled and sent to the *AddABoxData* WebGenesis service:

```
# Initialize data dictionary
data_dict = {}

# Add location
data_dict["location_" + incident_id] = {
    "type": "Location",
    "properties": {
        "latitude": position_latitude,
        "longitude": position_longitude
    }
}

# Add incident report
data_dict["incident_report_" + incident_id] = {
    "type": "IncidentReport",
    "properties": {
        "hasReportID": incident_id,
        "hasReportLocation": "location_" + incident_id,
        "hasReportText": json.dumps(self.message, indent=3)
    }
}

# Initialize query dictionary
```

```
query_dict = {
    "data": data_dict,
    "defaultprefix": "http://beaware-project.eu/beAWARE/#"
}

self.insert_into_webgenesis(json.dumps(query_dict, indent=3))
```

Respectively, the *Reasoner.top021_incident_report()* function will be executed. It will alter the incoming message accordingly and forward it back to the MSB. More specifically, the topic name will be changed:

```
# Copy incoming message
outgoing_message = self.incoming_message

# Change topic name in header
outgoing_message['header']['topicName'] = "TOP101 INCIDENT REPORT"
```

Moreover, the reasoner will check if this new report is within the spatial range of previous incidents. In this case, it will also change the incident ID in order to place the report within the existing incident cluster on the PSAP map. This new ID will also be stored to the KB:

```
# Check if this incident is nearby previous incidents
psap_incident_id = self.calculate_psap_incident_id(incident_id, lat,
long)

# Insert the PSAP incident ID to the KB
self.webgenesis_client.set_incident_report_psap_id(incident_id,
psap_incident_id)

# Set the PSAP incident ID to the outgoing message
outgoing_message['body']['incidentID'] = psap_incident_id
```

Finally, the assembled outgoing message will be produced to the MSB:

```
# Produce outgoing message
self.produce_message(outgoing_message['header']['topicName'],
outgoing_message)
```

Now that the knowledge base contains a minimal set of instances which represent this specific incident report, all future incoming updates (attached media files, attachment analysis results, etc.) of the report, originating from multiple beAWARE components, will be used to enrich the stored knowledge with new instances and relationships. Indicatively, in case that an incident report update contains an attached image, the image analysis component will provide its results to the KBS in a form similar to this:

```
{
    "image": {
        "crisis_level": "severe",
        "width": 749,
        "height": 500,
        "crisis_type": "flood",
        "timestamp": "2017-10-13 09:12:42.519408",
        "name": "image_x",
```

```
    "target": [{
      "left": 336,
      "risk": 0.17402849342347693,
      "width": 129,
      "top": 67,
      "height": 326,
      "confidence": 0.984,
      "type": "person"
    },
    {
      "left": 613,
      "risk": 0.669205328161215,
      "width": 31,
      "top": 215,
      "height": 28,
      "confidence": 0.974,
      "type": "car"
    },
    {
      "left": 327,
      "risk": 0.877536324101336,
      "width": 95,
      "top": 21,
      "height": 144,
      "confidence": 0.838,
      "type": "person"
    }
  ]
}
```

This sample output indicates that two persons and a car were discovered in a flooded location, with corresponding risk and confidence scores. The KBS will parse the results and will populate the appropriate classes shown in Figure 5 (**MediaItem**, **DataAnalysis**, **Detection**, **Incident**, **VulnerableObject**, etc.). Moreover, the following properties will be instantiated accordingly:

- **has attachment** – Associates an incident report to an attached media item (image, video, audio, text or HTML file).
- **relates to task** – When a media item is received, an analysis task is performed by the corresponding system component (image analysis, video analysis, etc.). This property will relate the media item with this task.
- **task produces dataset** – Associates the analysis task to the dataset it produced.
- **contains detection** – An analysis dataset may contain multiple detections, which are stored using this property.
- **involves incident** – Certain analysis components recognize the major incident (e.g. fire, flood, etc.) portrayed in a media item.
- **detects incident** – In some cases, a detection (and not the whole dataset, as above) detects an incident (e.g. traffic, collapse, etc).

- **detects participant** – In other cases, a detection might detect a vulnerable object (e.g. person, vehicle, building, etc.) as a participant.
- **involves participant** – This property associates a vulnerable object that has been found to participate in a detected incident.

Once all relevant data are populated in the ontology, the KBS will perform semantic reasoning (see next subsection) to infer new knowledge, and will, then, update the user interface component (PSAP) for this incident report. This will happen with the production of a new message in topic *TOP101_INCIDENT_REPORT*.

It is worth mentioning that the interaction of the KBS with the triplestore is achieved through SPARQL select queries for knowledge retrieval, while the *addABoxData* service of WebGenesis is used for data insertion.

4.2 Semantic Reasoning

Besides monitoring the message bus and semantically integrating information, the KBS incorporates a semantic reasoning mechanism to infer underlying knowledge and discover links between incidents during a crisis. This mechanism is rule-based, implemented with a combination of Python code and an elaborate SPARQL ruleset. The module is still in progress and will be further refined and presented in the upcoming deliverable D4.3, which is due M34. The following subsections briefly present some tasks handled by this reasoning mechanism.

4.2.1 Calculation of Incident Severity Index

Based on user requirements, when human beings are involved in an incident of high risk (fire, flood, etc.), this incident should be classified as severe, in order to attract attention during the decision making process. This calculation is implemented via SPARQL as follows:

```
DELETE { ?incident :hasIncidentSeverity ?previous_severity }
INSERT { ?incident :hasIncidentSeverity "severe" }
WHERE {
    ?incident rdf:type :Incident .
    { ?incident :isOfIncidentType :flooding }
    UNION
    { ?incident :isOfIncidentType :fire } .
    ?participant :participantIsInvolvedIn ?incident .
    ?participant rdf:type :Population .
    OPTIONAL {
        ?incident :hasIncidentSeverity ?previous_severity .
    }
}
```

4.2.2 Spatial Clustering of Incidents

The beAWARE system records incidents using a variety of sources (dedicated application, social media, etc.). Certain recordings tend to refer to the same incident, thus need to be clustered based on their location (latitude and longitude). The reasoning mechanism uses SPARQL to retrieve the exact position of all recorded incidents:

```
SELECT ?incident ?latitude ?longitude
WHERE {
    ?incident rdf:type :Incident .
    ?incident :hasLocation ?location .
    ?location :latitude ?latitude .
    ?location :longitude ?longitude .
}
```

Consequently, a piece of Python code classifies them in groups within a certain radius:

```
# For each existing incident
for existing_incident in existing_incidents:

    if existing_incident['lat'] != "unknown" and
        existing_incident['long'] != "unknown":

        # Calculate the distance from the given location
        distance = self.calculate_distance_between_locations(lat,
            long, existing_incident['lat'],
            existing_incident['long'])

        if distance < min_distance_from_existing_incident:
            psap_incident_id = existing_incident['psap_id']
            min_distance_from_existing_incident = distance

        # If a previous psap incident was found closer than 500m
        if min_distance_from_existing_incident <= 500:
            return psap_incident_id

        # Else, the incident should not be grouped with any previous
        # psap incident and create a new group
        else:
            return report_id
```

If the adopted triplestore provides support for GeoSPARQL [Perry & Herring, 2012], this functionality can be implemented by a more complex SPARQL query without the use of Python, with built-in geolocation functions such as **geof:distance**. In any case, this clustering process is intended to be enriched with other semantic criteria, other than location, such as temporal information and the taxonomy of incidents. This direction will be pursued in the final version of the ontology and the KBS component.

4.2.3 Monitoring of Safe Locations

beAWARE supports the initialization of safe locations and relief spots, mainly in the form of structures (buildings, parks, etc.) and infrastructure (streets, bridges, etc.). During a crisis,

citizens are notified through the dedicated application about the existence of such locations, safe detours and more. The reasoner is responsible for inferring the availability of these spots, using incoming semantics, and determining optimal alternatives in case of low availability levels. For example, if a bridge is reported to have collapsed during a flood crisis, the closest and safest river passage should be calculated and announced to the citizens.

4.3 Chapter Summary

This chapter presented the Knowledge Base Service (KBS) component that handles the vital task of semantic integration and reasoning in the beAWARE system. The KBS lies in the middle of all system communications, storing knowledge in the WebGenesis triplestore, and forwarding semantically enriched incident report messages towards other components. New knowledge is inferred through reasoning, while incidents and attachments are aggregated using spatial criteria.

5 Conclusions and Next Steps

This deliverable reported on the work conducted within tasks T4.3-4.5 and, more specifically, it presented the following key contributions:

- The **first iteration of the beAWARE ontology**, which is a lightweight knowledge representation model for semantically representing notions pertinent to the project. Contrary to existing third-party ontologies for crisis management, which share the drawback of covering only a subset of the notions involved in climate-related crisis management, the beAWARE ontology delivers a more holistic, all-around model for semantically representing: (a) climate-related natural disasters, (b) analysis of data from the multimodal sensors, and, (c) rescue unit assignments.
- The **Knowledge Base Repository (KBR)** that hosts the ontology, served by WebGenesis, a powerful and highly customizable content management system for ontology management and maintenance developed by beAWARE partner IOSB.
- The **Knowledge Base Service (KBS)**, the front-end service of the KB, which provides reasoning services to the beAWARE platform and utilizes the KBR and the rest of the system modules. The KBS is responsible for semantically integrating the multimodal input from other modules into the ontology. It also forwards the high-level knowledge inferred from the semantic reasoning process to other interested system components, like the report generator module and the PSAP.

The above components will be further refined during the second half of the beAWARE project timeframe, which also includes the execution of the pilot trials. The final version will be presented in D4.3 which is due M34. The following directions for improvements are foreseen:

- **Extension of the beAWARE ontology**, in order to cover more extensively aspects that are not adequately covered yet, like e.g. the semantic representation of rescue units and mission assignments (see subsection 2.3.3).
- **Implementation of ontology population techniques** for semantically enriching the beAWARE ontology with information from external sources, like e.g. DBpedia. This thread will be based on previous work of ours [Mitziyas et al., 2016; Kontopoulos et al., 2017].
- **Mapping beAWARE ontology constructs to other existing models**. This has not been implemented yet, since the ontology is still evolving, but such mappings will be integrated in the final iteration of the ontology, in order to establish semantic interoperability with other third-party solutions. This process typically involves the definition of ontology mappings in a separate ontology document that contains the mappings between the beAWARE ontology concepts and those of third-party vocabularies. This document facilitates the direct alignment and easy comprehension of terms, data and relations from multiple domains.

- **Improvements to the existing set of semantic reasoning rules**, like e.g. incident clustering based on criteria other than spatial (see subsection 4.2.2).
- **Extension of the semantic reasoning ruleset** with new and more elaborate rules for providing more meaningful assistance to the decision making processes. The pilot trials will play a significant role in expanding the ruleset.
- **Integration of more advanced semantic reasoning techniques**, including uncertainty and/or probabilistic reasoning methodologies, like e.g. fuzzy ontologies [Bobillo & Straccia, 2011] and defeasible reasoning [Garcia & Simari, 2004].
- **Revision of geographic query languages** and their implementation into WebGenesis to facilitate location-dependant information retrieval.

References

- de la Asunción, M., Castillo, L., Fdez-Olivares, J., García-Pérez, Ó., González, A., & Palao, F. (2005). SIADEx: An interactive knowledge-based planner for decision support in forest fire fighting. *AI Communications*, 18(4), 257-268.
- Babitski, G., Bergweiler, S., Grebner, O., Oberle, D., Paulheim, H., & Probst, F. (2011, May). SoKNOS—using semantic technologies in disaster management software. In *Extended Semantic Web Conference* (pp. 183-197). Springer, Berlin, Heidelberg.
- Babitski, G., Probst, F., Hoffmann, J., & Oberle, D. (2009). Ontology Design for Information Integration in Disaster Management. *GI Jahrestagung*, 154, 3120-3134.
- Bechhofer, S. (2009). OWL: Web ontology language. *Encyclopedia of Database Systems*, pp. 2008-2009. Springer US.
- Bernaras, A., Laresgoiti, I. and Corera, J. (1996). Building and reusing ontologies for electrical network applications. *Proceedings of the European Conf. on Artificial Intelligence (ECAI'96)*, pp. 298-302.
- Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The semantic web. *Scientific american*, 284(5), 34-43.
- Bobillo, F., & Straccia, U. (2011). Fuzzy ontology representation using OWL 2. *International Journal of Approximate Reasoning*, 52(7), 1073-1094.
- Couturier, M., & Wilkinson, E. (2005, April). Open advanced system for improved crisis management (OASIS). In *Proceedings of the 2nd International Information Systems for Crisis Response and Management (ISCRAM) Conference*, Brussels, Belgium.
- Falco, R., Gangemi, A., Peroni, S., Shotton, D., & Vitali, F. (2014, May). Modelling OWL ontologies with Graffoo. In *European Semantic Web Conference* (pp. 320-325). Springer, Cham.
- Fernandez, M., Gomez-Perez, A. and Juristo, N. (1997). METHONTOLOGY: From Ontological Art Towards Ontological Engineering. *AAAI Technical Report SS-97-06*, pp. 33-40.
- Gangemi, A., Catenacci, C., Ciaramita, M., & Lehmann, J. (2005). Ontology evaluation and validation: an integrated formal model for the quality diagnostic task. *Technical Report*. Laboratory for Applied Ontology, ISTC-CNR, Rome/Trento, Italy.
- Gangemi, A. and Presutti, V. (2009). Ontology design patterns. *Handbook on ontologies*, pp. 221-243. Springer Berlin Heidelberg.
- Garcia, A., & Simari, G. (2004). Defeasible logic programming: An argumentative approach. *Theory and practice of logic programming*, 4(1+2):95–138, 2004.
- Gruber, T. (1993). A translational approach to portable ontologies. *Knowledge Acquisition*, 5(2), 199-229.
- Grüninger, M. and Fox, M.S. (1995). Methodology for the design and evaluation of ontologies. In Skuce D (ed) *IJCAI95 Workshop on Basic Ontological Issues in Knowledge Sharing*, pp 6.1–6.10.
- Harris, S., Seaborne, A. and Prud'hommeaux, E. (2013), *SPARQL 1.1 query language, W3C recommendation*, 21(10).
- Hendler, J. (2009). Web 3.0 Emerging. *Computer*, 42(1).
- Jarrar, M. and Meersman, R. (2008). Ontology Engineering - The DOGMA Approach. *Advances in Web Semantics I*, LNCS Vol. 4891, pp. 7-34.
- Kontopoulos, E. Mitzi, P. Moßgraber, J. Hertweck, P. van der Schaaf, H. Hilbring, D. Lombardo, F. Norbiato, D. Ferri, M. Karakostas, A. Vrochidis, S. and Kompatsiaris, I. (2018), Ontology-based Representation of Crisis Management Procedures for Climate Events, *1st Int. Workshop on Intelligent Crisis Management Technologies for climate events (ICMT), collocated with ISCRAM 2018*, to be presented.

- Kontopoulos, E., Mitzias, P., Riga, M., Kompatsiaris, I. (2017). A Domain-Agnostic Tool for Scalable Ontology Population and Enrichment from Diverse Linked Data Sources. In: Kalinichenko, L.A., Manolopoulos, Y., Skvortsov, N.A., and Sukhomlin, V.A. (eds.) *Selected Papers of the XIX International Conference on Data Analytics and Management in Data Intensive Domains (DAMDID/RCDL 2017)*. pp. 184–190. CEUR Workshop Proceedings Vol 2022, Moscow, Russia.
- Lauras, M., Truptil, S., & Bénaben, F. (2015). Towards a better management of complex emergencies through crisis management meta-modelling. *Disasters*, 39(4), 687-714.
- Lenat, D.B. and Guha, R.V. (1989). Building large knowledge-based systems; representation and inference in the Cyc project. Addison-Wesley Longman Publishing Co., Inc.
- Liang, S., Huang, C., and Khalafbeigi, T. (2016). *OGC Sensor Things API*. Open Geospatial Consortium – Wayland, MA, USA.
- Limbu, M. (2012). Management of a Crisis (MOAC) Vocabulary Specification. Available online: <http://www.observedchange.com/moac/ns/>, last accessed: Apr'18.
- Liu, S., Brewster, C., & Shaw, D. (2013). Ontologies for crisis management: a review of state of the art in ontology design and usability. In: *Proceedings of the 10th International ISCRAM Conference* – Baden-Baden, Germany, May 2013, pp. 349–359.
- Liu, Y., Chen, S., & Wang, Y. (2014). SOFERS: Scenario Ontology for Emergency Response System. *JNW*, 9(9), 2529-2535.
- Mescherin, S. A., Kirillov, I., & Klimenko, S. (2013, October). Ontology of emergency shared situation awareness and crisis interoperability. In *Cyberworlds (CW), 2013 International Conference on* (pp. 159-162). IEEE.
- Miles, A., & Bechhofer, S. (2009). *SKOS simple knowledge organization system reference*. W3C recommendation, 18, W3C.
- Mitzias, P., Riga, M., Kontopoulos, E., Stavropoulos, T.G., Andreadis, S., Meditskos, G., Kompatsiaris, I. (2016). User-Driven Ontology Population from Linked Data Sources. In: *7th Int. Conf. on Knowledge Engineering and the Semantic Web (KESW 2016)*. pp. 31–41. Springer International Publishing, Prague, Czech Republic.
- Musen, M.A. (2015). The Protégé project: A look back and a look forward. *AI Matters. Association of Computing Machinery Specific Interest Group in Artificial Intelligence*, 1(4), June. DOI: 10.1145/2557001.25757003.
- Norbiato, D., Castro, C., Vourvachis, I., Janniche, A., Karppinen, A., Karakostas, A., & Ferri, M (2017). D2.1 Use cases and initial user requirements. beAWARE H2020 project deliverable. Available online: http://beaware-project.eu/wp-content/uploads/2017/07/D2.1_beAWARE.pdf, last accessed: Jun'18.
- OGC_1 Open Geospatial Consortium (2017). Observations and Measurements, ISO/DIS 19156, <http://www.opengeospatial.org/standards/om, April 2017>.
- OGC_2 Open Geospatial Consortium. SensorThings API Part 1: Sensing, <http://www.opengeospatial.org/standards/sensorthings, April 2017>.
- Perry, M., & Herring, J. (2012). OGC GeoSPARQL-A geographic query language for RDF data. *OGC implementation standard*.
- Pinto, H.S., Staab, S. and Tempich, C. (2004). DILIGENT: Towards a fine-grained methodology for Distributed, Loosely-controlled and evolvinG Engineering of oNTologies. *Proceedings of the 16th European Conf. on Artificial Intelligence (ECAI)*, pp. 393-397.
- Poveda-Villalón, M., Gómez-Pérez, A., & Suárez-Figueroa, M. C. (2014). Oops!(ontology pitfall scanner!): An on-line tool for ontology evaluation. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 10(2), 7-34.
- Rospocher, M., & Serafini, L. (2012, December). An ontological framework for decision support. In *Joint International Semantic Technology Conference* (pp. 239-254). Springer, Berlin, Heidelberg.

- Suárez-Figueroa, M., Gómez-Pérez, A. and Villazón-Terrazas, B. (2009). How to write and use the Ontology Requirements Specification Document. *On the move to meaningful internet systems: OTM 2009*. Part of the Lecture Notes in Computer Science book series (LNCS, Vol. 5871), 966-982.
- Sure, Y., Staab, S. and Studer, R. (2004). On-To-Knowledge Methodology (OTKM). *Handbook on Ontologies*, pp. 117-132.
- Swartout, B., Ramesh, P., Knight, K. and Russ, T. (1997). Towards distributed use of large-scale ontologies. *Symposium on Ontological Engineering of AAAI*, pp. 138-148.
- Tartir, S., Arpinar, I. B., & Sheth, A. P. (2010). Ontological evaluation and validation. In *Theory and applications of ontology: Computer applications* (pp. 115-130). Springer Netherlands.
- Truptil, S., Bénaben, F., Couget, P., Lauras, M., Chapurlat, V., & Pingaud, H. (2008). Interoperability of information systems in crisis management: Crisis modeling and metamodeling. *Enterprise Interoperability III*, 583-594.
- Uschold, M. and King, M. (1995). Towards methodology for building ontologies. *Workshop on Basic Ontological Issues in Knowledge Sharing, held in Conjunction with IJCAI-95*. Cambridge, UK.
- [W3C, 2012] *OWL 2 Web Ontology Language Document Overview (Second Edition)*. W3C Recommendation 11 December 2012, available online: <http://www.w3.org/TR/owl2-overview/>
- Zavarella, V., Tanev, H., Steinberger, R., & Van der Goot, E. (2014). An Ontology-Based Approach to Social Media Mining for Crisis Management. In *SSA-SMILE@ ESWC* (pp. 55-66).
- Zemmouchi-Ghomari, L., & Ghomari, A. R. (2013). Translating natural language competency questions into SPARQL Queries: a case study. In *1st Int. Conf. on Building and Exploring Web Based Environments*, pp. 81-86.

Appendix A – Ontology Specification

This appendix lists the ontology classes, object properties and data properties.

Classes

Agriculture

Name	Agriculture
Definition	Agricultural used spaces (e.g. field and acre).
Subclass of	EcologicalAsset

Animal

Name	Animal
Definition	Represents animals in danger during a natural disaster.
Subclass of	LivingBeing

Asset

Name	Asset
Definition	Any non-living item of interest.
Subclass of	VulnerableObject
Disjoint with	LivingBeing

Audio Item

Name	AudioItem
Definition	Represents an audio recording.
Subclass of	AudioItem
Disjoint with	TextItem, ImageItem, VideoItem

Bridge

Name	Bridge
Definition	Represents bridges.
Subclass of	Structure

Building

Name	Building
Definition	A structure with walls and a roof (e.g. a house or factory).
Subclass of	Structure

Car

Name	Car
Definition	Represents any type of car.
Subclass of	Vehicle

Climate Parameter

Name	ClimateParameter
Definition	Represents the actual manifestation of a climate parameter measurement.
Instance of	owl:Class

Climate Parameter Type

Name	ClimateParameterType
Definition	Represents various types of climate-related parameters (e.g. temperature, humidity, etc.).
Instance of	owl:Class

Communication

Name	Communication
Definition	Represents any type of (tele)communication infrastructure.
Subclass of	Infrastructure

Data Analysis

Name	DataAnalysis
Definition	A type of a task involving data analysis.
Subclass of	Task

Dataset

Name	Dataset
Definition	Represents the dataset produced by some task.
Instance of	owl:Class

Detection

Name	Detection
Definition	Represents detections in a dataset (e.g. a person trapped in the flood).
Instance of	owl:Class

Dunes

Name	Dunes
Definition	Represents dunes.
Subclass of	EcologicalAsset

Ecological Asset

Name	EcologicalAsset
Definition	Ecological assets of various types.
Subclass of	Asset

Educational Facility

Name	Educational Facility
Definition	Represents any type of educational facility (e.g. school or university).
Subclass of	Infrastructure

Electric Energy Supply

Name	ElectricEnergySupply
Definition	Represents electric energy supply infrastructure.
Subclass of	Energy

Energy

Name	Energy
Definition	Represents any type of energy-generating infrastructure.
Subclass of	Infrastructure

Fire Department

Name	FireDepartment
Definition	Represents fire departments.
Subclass of	Infrastructure

Garbage Collection

Name	GarbageCollection
Definition	Represents garbage collection infrastructure and services.
Subclass of	Infrastructure

Hospital

Name	Hospital
Definition	Represents hospitals.
Subclass of	Infrastructure

Human

Name	Human
Definition	Represents human beings in danger.
Subclass of	Animal

Image Analysis

Name	ImageAnalysis
Definition	The task of analyzing images.
Subclass of	DataAnalysis
Disjoint with	TextAnalysis, VideoAnalysis

Image Item

Name	ImageItem
Definition	Represents a captured image.
Subclass of	ImageItem
Disjoint with	TextItem, AudioItem, VideoItem

Impact

Name	Impact
Definition	Represents the impact of natural disasters and incidents.
Instance of	owl:Class

Impact Type

Name	ImpactType
Definition	Represents the various types of impacts, like e.g. injuries, damage to properties etc.
Instance of	owl:Class

Incident

Name	Incident
Definition	Represents the various incidents taking place during a natural disaster.
Instance of	owl:Class

Incident Type

Name	IncidentType
Definition	Represents the various types of incidents, like e.g. floods, blocked streets etc.
Instance of	owl:Class

Infrastructure

Name	Infrastructure
Definition	Represents critical infrastructure that is in danger during a natural disaster.
Subclass of	Asset

Levee

Name	Levee
Definition	An embankment for preventing flooding.
Subclass of	Structure

Livestock

Name	Livestock
Definition	Represents domestic animals.
Subclass of	Animal

Living Being

Name	LivingBeing
Definition	Any living being that is in danger during a natural disaster.
Subclass of	VulnerableObject
Disjoint with	Asset

Location

Name	Location
Definition	Represents a location (point or area), indicated by latitude, longitude, and radius.
Instance of	owl:Class

Media Item

Name	MediaItem
Definition	Represents a generic media item. Subclasses include specific types of media items.
Instance of	owl:Class

Mission

Name	Mission
Definition	Represents a mission assigned to a rescue unit during a crisis.
Instance of	owl:Class

Monument

Name	Monument
Definition	A structure or building that is built to honour a special person or event.
Subclass of	Structure

Natural Disaster

Name	NaturalDisaster
Definition	Represents the actual manifestation of a natural disaster type. An instance of a natural disaster has specific climate conditions with specific values (e.g. temperature = 45) plus some other properties (e.g. start/end time).
Instance of	owl:Class

Natural Disaster Type

Name	NaturalDisasterType
Definition	Represents the various types of disasters, like e.g. floods, forest fires, storms or earthquakes etc.
Instance of	owl:Class

Natural Habitat

Name	NaturalHabitat
Definition	Represents natural habitats.
Subclass of	EcologicalAsset

Plant

Name	Plant
Definition	Represents the fauna.
Subclass of	EcologicalAsset

Police

Name	Police
Definition	Represents law enforcement infrastructure and services.
Subclass of	Infrastructure

Property

Name	Property
Definition	Represents any type of private property.
Subclass of	Asset

Public Transportation

Name	PublicTransportation
Definition	Represents public transportation services and infrastructure.
Subclass of	Transportation

Responder

Name	Responder
Definition	Represents a first responder unit (e.g. a firefighter, police officer or emergency medical physician).
Instance of	owl:Class

River

Name	River
Definition	Represents rivers.
Subclass of	EcologicalAsset

Sensor

Name	Sensor
Definition	A Sensor is an instrument that observes a property or phenomenon with the goal of producing an estimate of the value of a parameter.
Instance of	owl:Class

Sewer

Name	Sewer
Definition	Represents sewage infrastructure.
Subclass of	Infrastructure

Square

Name	Square
Definition	Represents squares in danger during an environmental crisis.
Subclass of	Structure

Street

Name	Street
Definition	Represents the road network infrastructure.
Subclass of	Transportation

Structure

Name	Structure
Definition	Represents various structures and buildings.
Subclass of	Asset

Subway

Name	Subway
Definition	Represents subway infrastructure.
Subclass of	PublicTransportation

Task

Name	Task
Definition	A task that has to do with analyzing or processing items.
Instance of	owl:Class

Text Analysis

Name	TextAnalysis
Definition	The task of analyzing textual corpora.
Subclass of	DataAnalysis
Disjoint with	ImageAnalysis, VideoAnalysis

Text Item

Name	TextItem
Definition	Represents a piece of text.
Subclass of	TextItem
Disjoint with	AudioItem, ImageItem, VideoItem

Transportation

Name	Transportation
Definition	Represents transportation services and infrastructure.
Subclass of	Infrastructure

Vehicle

Name	Vehicle
Definition	Represents any type of vehicle.
Subclass of	Property

Video Analysis

Name	VideoAnalysis
Definition	The task of analyzing videos.
Subclass of	DataAnalysis
Disjoint with	TextAnalysis, VideoAnalysis

Video Item

Name	VideoItem
Definition	Represents a video recording.
Subclass of	VideoItem
Disjoint with	AudioItem, ImageItem, TextItem

Vulnerable Object

Name	VulnerableObject
Definition	Any living being or object that needs to be protected from hazards.
Instance of	owl:Class

Wall

Name	Wall
Definition	A vertical structure often made of stone or brick that divides or surrounds something.
Subclass of	Structure

Water Supply

Name	WaterSupply
Definition	Represents water supply and sanitation infrastructure.
Subclass of	Infrastructure

Wildlife

Name	Wildlife
Definition	Represents wild animals.
Subclass of	Animal

Object Properties

caused by disaster

Name	causedByDisaster
Definition	Indicates which disaster type is responsible for this impact type.
Instance of	owl:ObjectProperty
Domain	ImpactType
Range	NaturalDisasterType
Inverse of	causesDisasterImpact

caused by incident

Name	causedByIncident
Definition	Indicates which incident type has caused this impact type.
Instance of	owl:ObjectProperty
Domain	ImpactType
Range	IncidentType
Inverse of	causesIncidentImpact

causes disaster impact

Name	causesDisasterImpact
Definition	Indicates which impact type is caused by this natural disaster type.
Instance of	owl:ObjectProperty
Domain	NaturalDisasterType
Range	ImpactType
Inverse of	causedByDisaster

causes incident impact

Name	causesIncidentImpact
Definition	Indicates which impact type is caused by this type of incident.
Instance of	owl:ObjectProperty
Domain	IncidentType
Range	ImpactType
Inverse of	causedByIncident

characterized by parameter type

Name	characterizedByParameterType
Definition	Associates a natural disaster type with a climate parameter type.
Instance of	owl:ObjectProperty
Domain	NaturalDisasterType
Range	ClimateParameterType
Inverse of	characterizesDisasterType

characterizes disaster type

Name	characterizesDisasterType
Definition	Associates a climate parameter type with a natural disaster type.
Instance of	owl:ObjectProperty
Domain	ClimateParameterType
Range	NaturalDisasterType
Inverse of	characterizedByParameterType

contained in dataset

Name	containedInDataset
Definition	Indicates the dataset this detection object is contained in.
Instance of	owl:ObjectProperty
Domain	Detection
Range	Dataset
Inverse of	containsDetection

contains detection

Name	containsDetection
Definition	A dataset may contain multiple detections.
Instance of	owl:ObjectProperty
Domain	Dataset
Range	Detection
Inverse of	containedInDataset

derived from

Name	derivedFrom
Definition	Represents the fact that an incident may lead to other incidents.
Instance of	owl:TransitiveProperty
Domain	Incident
Range	Incident

detects incident

Name	detectsIncident
Definition	Indicates which incidents were detected by a specific detection activity (e.g. a flooding incident, a blocked road etc.).
Instance of	owl:ObjectProperty
Domain	Detection
Range	Incident
Inverse of	incidentDetectedBy

detects participant

Name	detectsParticipant
Definition	Indicates which participants were detected by a specific detection activity (e.g. humans in danger, submerged cars etc.).
Instance of	owl:ObjectProperty
Domain	Detection
Range	VulnerableObject
Inverse of	participantDetectedBy

has climate parameter measurement

Name	hasClimateParameterMeasurement
Definition	Associates a natural disaster occurrence to a climate parameter measurement.
Instance of	owl:ObjectProperty
Domain	NaturalDisaster
Range	ClimateParameter
Inverse of	relatesToNaturalDisaster

has climate parameter occurrence

Name	hasClimateParameterOccurrence
Definition	Assigns a climate parameter type to a specific climate parameter.
Instance of	owl:ObjectProperty
Domain	ClimateParameterType
Range	ClimateParameter
Inverse of	isOfClimateParameterType

has disaster location

Name	hasDisasterLocation
Definition	Represents the location where a natural disaster takes place.
Instance of	owl:ObjectProperty
Domain	NaturalDisaster
Range	Location
Inverse of	isLocationOfDisaster

has disaster occurrence

Name	hasDisasterOccurrence
Definition	Links a natural disaster type to a natural disaster.
Instance of	owl:ObjectProperty
Domain	NaturalDisasterType
Range	NaturalDisaster
Inverse of	isOfDisasterType

has impact occurrence

Name	hasImpactOccurrence
Definition	Links an impact type to a specific impact occurrence.
Instance of	owl:ObjectProperty
Domain	ImpactType
Range	Impact
Inverse of	isOfImpactType

has impact on

Name	hasImpactOn
Definition	Indicates which vulnerable object is impacted by the specific impact occurrence.
Instance of	owl:ObjectProperty
Domain	Impact
Range	VulnerableObject
Inverse of	isImpactedBy

has incident climate parameter

Name	hasIncidentClimateParameter
Definition	Associates an incident to a climate parameter measurement.
Instance of	owl:ObjectProperty
Domain	Incident
Range	ClimateParameter
Inverse of	isClimateParameterOfIncident

has incident impact

Name	hasIncidentImpact
Definition	Indicates the impact of the specific incident.
Instance of	owl:ObjectProperty
Domain	Incident
Range	Impact
Inverse of	isImpactOfIncident

has incident location

Name	hasIncidentLocation
Definition	Represents the location where an incident occurs.
Instance of	owl:ObjectProperty
Domain	Incident
Range	Location
Inverse of	isLocationOfIncident

has incident occurrence

Name	hasIncidentOccurrence
Definition	Links an incident type to an incident.
Instance of	owl:ObjectProperty
Domain	IncidentType
Range	Incident
Inverse of	isOfIncidentType

has measurement location

Name	hasMeasurementLocation
Definition	Represents the location where a climate parameter measurement was taken.
Instance of	owl:ObjectProperty
Domain	ClimateParameter
Range	Location
Inverse of	isLocationOfMeasurement

has media location

Name	hasMediaLocation
Definition	Represents the location where a media item was captured.
Instance of	owl:ObjectProperty
Domain	MediaItem
Range	Location
Inverse of	isLocationOfMediaItem

has related incident

Name	hasRelatedIncident
Definition	Relates an incident to a natural disaster.
Instance of	owl:ObjectProperty
Domain	NaturalDisaster
Range	Incident

has report location

Name	hasReportLocation
Definition	Indicates the location of the incident report.
Instance of	owl:ObjectProperty
Domain	IncidentReport
Range	Location
Inverse of	isLocationOfReport

has responder location

Name	hasResponderLocation
Definition	Represents the location where a rescue unit is at.
Instance of	owl:ObjectProperty
Domain	Responder
Range	Location
Inverse of	isLocationOfResponder

incident detected by

Name	incidentDetectedBy
Definition	Indicates the detection activity that detected this incident.
Instance of	owl:ObjectProperty
Domain	Incident
Range	Detection
Inverse of	detectsIncident

involved in dataset

Name	involvedInDataset
Definition	Indicates the dataset this incident is involved in.
Instance of	owl:ObjectProperty
Domain	Incident
Range	Dataset
Inverse of	involvesIncident

involved in incident

Name	involvedInIncident
Definition	Indicates the incident a vulnerable object is involved in.
Instance of	owl:ObjectProperty
Domain	VulnerableObject
Range	Incident
Inverse of	involvesParticipant

involves incident

Name	involvesIncident
Definition	Certain analysis components recognize the major incident (e.g. fire, flood, etc.) portrayed in a media item.
Instance of	owl:ObjectProperty
Domain	Dataset
Range	Incident
Inverse of	involvedInDataset

involves participant

Name	involvesParticipant
Definition	Indicates the vulnerable objects involved in an incident.
Instance of	owl:ObjectProperty
Domain	Incident
Range	VulnerableObject
Inverse of	involvedInIncident

is assigned mission

Name	isAssignedMission
Definition	Indicates the mission assignment of a rescue unit.
Instance of	owl:ObjectProperty
Domain	Responder
Range	Mission
Inverse of	isAssignedToResponder

is assigned to responder

Name	isAssignedToResponder
Definition	Links a mission to a rescue unit.
Instance of	owl:ObjectProperty
Domain	Mission
Range	Responder
Inverse of	isAssignedMission

is climate parameter of incident

Name	isClimateParameterOfIncident
Definition	Links a climate parameter measurement to an incident.
Instance of	owl:ObjectProperty
Domain	ClimateParameter
Range	Incident
Inverse of	hasIncidentClimateParameter

is impact of incident

Name	isImpactOfIncident
Definition	Links an impact to an incident.
Instance of	owl:ObjectProperty
Domain	Impact
Range	Incident
Inverse of	hasIncidentImpact

is impacted by

Name	isImpactedBy
Definition	Links an affected object to an impact.
Instance of	owl:ObjectProperty
Domain	VulnerableObject
Range	Impact
Inverse of	hasImpactOn

is location of disaster

Name	isLocationOfDisaster
Definition	Links a location to a natural disaster occurrence.
Instance of	owl:ObjectProperty
Domain	Location
Range	NaturalDisaster
Inverse of	hasDisasterLocation

is location of incident

Name	isLocationOfIncident
Definition	Links a location to an incident occurrence.
Instance of	owl:ObjectProperty
Domain	Location
Range	Incident
Inverse of	hasIncidentLocation

is location of measurement

Name	isLocationOfMeasurement
Definition	Links a location to a climate parameter measurement.
Instance of	owl:ObjectProperty
Domain	Location
Range	ClimateParameter
Inverse of	hasMeasurementLocation

is location of media item

Name	isLocationOfMediaItem
Definition	Links a location to a media item.
Instance of	owl:ObjectProperty
Domain	Location
Range	MediaItem
Inverse of	hasMediaLocation

is location of report

Name	isLocationOfReport
Definition	Links an incident report to a location.
Instance of	owl:ObjectProperty
Domain	Location
Range	IncidentReport
Inverse of	hasReportLocation

is location of responder

Name	isLocationOfResponder
Definition	Links a location to a rescue unit.
Instance of	owl:ObjectProperty
Domain	Location
Range	Responder
Inverse of	hasResponderLocation

is of climate parameter type

Name	isOfClimateParameterType
Definition	Associates a climate parameter with a respective type.
Instance of	owl:ObjectProperty
Domain	ClimateParameter
Range	ClimateParameterType
Inverse of	hasClimateParameterOccurrence

is of disaster type

Name	isOfDisasterType
Definition	Links a natural disaster to a natural disaster type.
Instance of	owl:ObjectProperty
Domain	NaturalDisaster
Range	NaturalDisasterType
Inverse of	hasDisasterOccurrence

is of impact type

Name	isOfImpactType
Definition	Links an impact to an impact type.
Instance of	owl:ObjectProperty
Domain	Impact
Range	ImpactType
Inverse of	hasImpactOccurrence

is of incident type

Name	isOfIncidentType
Definition	Links an incident to an incident type.
Instance of	owl:ObjectProperty
Domain	Incident
Range	IncidentType
Inverse of	hasIncidentOccurrence

is result of

Name	isResultOf
Definition	Represents the fact that a natural disaster can be the result of other natural disasters.
Instance of	owl:TransitiveProperty
Domain	NaturalDisasterType
Range	NaturalDisasterType
Inverse of	leadsTo

leads to

Name	leadsTo
Definition	Represents the fact that a natural disaster can lead to other natural disasters.
Instance of	owl:TransitiveProperty
Domain	NaturalDisasterType
Range	NaturalDisasterType
Inverse of	isResultOf

observes

Name	observes
Definition	The sensor observes a specific parameter.
Instance of	owl:ObjectProperty
Domain	Sensor
Range	ClimateParameter

participant detected by

Name	participantDetectedBy
Definition	Indicates which detection activity detected the participants.
Instance of	owl:ObjectProperty
Domain	VulnerableObject
Range	Detection
Inverse of	detectsParticipant

produced by task

Name	producedByTask
Definition	Indicates the task a dataset was produced from.
Instance of	owl:ObjectProperty
Domain	Dataset
Range	Task
Inverse of	taskProducesDataset

relates to

Name	relatesTo
Definition	Associates media items.
Instance of	owl:SymmetricProperty
Domain	MediaItem
Range	MediaItem

relates to incident

Name	relatesToIncident
Definition	Associates a mission to an incident occurrence.
Instance of	owl:ObjectProperty
Domain	Mission
Range	Incident
Inverse of	relatesToMission

relates to media item

Name	relatesToMediaItem
Definition	Associates a task to a media item.
Instance of	owl:ObjectProperty
Domain	Task
Range	MediaItem
Inverse of	relatesToTask

relates to mission

Name	relatesToMission
Definition	Associates an incident occurrence to a mission.
Instance of	owl:ObjectProperty
Domain	Incident
Range	Mission
Inverse of	relatesToIncident

relates to natural disaster

Name	relatesToNaturalDisaster
Definition	Associates a climate parameter measurement to a natural disaster occurrence.
Instance of	owl:ObjectProperty
Domain	ClimateParameter
Range	NaturalDisaster
Inverse of	hasClimateParameterMeasurement

relates to task

Name	relatesToTask
Definition	Associates a media item to a task.
Instance of	owl:ObjectProperty
Domain	MediaItem
Range	Task
Inverse of	relatesToMediaItem

sensor produces dataset

Name	sensorProducesDataset
Definition	Indicates the dataset(s) produced by this sensor.
Instance of	owl:ObjectProperty
Domain	Sensor
Range	Dataset

task produces dataset

Name	taskProducesDataset
Definition	Indicates the dataset produced by this task.
Instance of	owl:ObjectProperty
Domain	Task
Range	Dataset
Inverse of	producedByTask

Data Properties**belongs to collection**

Name	belongsToCollection
Definition	Links a media item to one or more collections.
Instance of	owl:DatatypeProperty
Domain	MediaItem
Range	xsd:string

has analyzed media source

Name	hasAnalyzedMediaSource
Definition	Links a media item to a URI pointing to the analysed item (e.g. processed image or video source).
Instance of	owl:DatatypeProperty
Domain	MediaItem
Range	xsd:anyURI

has dataset results source

Name	hasDatasetResultsSource
Definition	Points to a URI with the dataset source.
Instance of	owl:DatatypeProperty
Domain	Dataset
Range	xsd:anyURI

has detection confidence

Name	hasDetectionConfidence
Definition	Represents the confidence score of a particular detection (e.g. an image analysis algorithm detects a trapped car in the flood with 80% confidence).
Instance of	owl:DatatypeProperty
Domain	Detection
Range	xsd:float

has detection end

Name	hasDetectionEnd
Definition	Indicates the end of the detection.
Instance of	owl:DatatypeProperty
Domain	Detection
Range	xsd:dateTime

has detection risk

Name	hasDetectionRisk
Definition	Indicates the risk level of the detection.
Instance of	owl:DatatypeProperty
Domain	Detection
Range	xsd:float

has detection start

Name	hasDetectionStart
Definition	Indicates the start of the detection.
Instance of	owl:DatatypeProperty
Domain	Detection
Range	xsd:dateTime

has detection timestamp

Name	hasDetectionTimestamp
Definition	Represents the timestamp of the detection.
Instance of	owl:DatatypeProperty
Domain	Detection
Range	xsd:dateTime

has incident end

Name	hasIncidentEnd
Definition	Indicates the end of the incident.
Instance of	owl:DatatypeProperty
Domain	Incident
Range	xsd:dateTime

has incident priority

Name	hasIncidentPriority
Definition	Indicates the priority of the incident (e.g. incidents involving humans in danger are of high priority).
Instance of	owl:DatatypeProperty
Domain	Incident
Range	xsd:string

has incident start

Name	hasIncidentStart
Definition	Indicates the start of the incident.
Instance of	owl:DatatypeProperty
Domain	Incident
Range	xsd:dateTime

has incident severity

Name	hasIncidentSeverity
Definition	Indicates the severity of the incident (e.g. incidents involving injured or dead people are of high severity).
Instance of	owl:DatatypeProperty
Domain	Incident
Range	xsd:string

has latitude

Name	lat
Definition	Geographic latitude coordinates of a location.
Instance of	owl:DatatypeProperty
Domain	Location
Range	xsd:float

has longitude

Name	long
Definition	Geographic longitude coordinates of a location.
Instance of	owl:DatatypeProperty
Domain	Location
Range	xsd:float

has measurement timestamp

Name	hasMeasurementTimestamp
Definition	Indicates the timestamp of a climate parameter measurement.
Instance of	owl:DatatypeProperty
Domain	ClimateParameter
Range	xsd:dateTime

has media item timestamp

Name	hasMediaItemTimestamp
Definition	Indicates the timestamp when a media item was created.
Instance of	owl:DatatypeProperty
Domain	MediaItem
Range	xsd:dateTime

has mission end

Name	hasMissionEnd
Definition	Indicates the end of the rescue mission.
Instance of	owl:DatatypeProperty
Domain	Mission
Range	xsd:dateTime

has mission priority

Name	hasMissionPriority
Definition	Indicates the priority of the rescue mission.
Instance of	owl:DatatypeProperty
Domain	Mission
Range	xsd:string

has mission start

Name	hasMissionStart
Definition	Indicates the start of the rescue mission.
Instance of	owl:DatatypeProperty
Domain	Mission
Range	xsd:dateTime

has mission status

Name	hasMissionStatus
Definition	Indicates the current status of the rescue mission.
Instance of	owl:DatatypeProperty
Domain	Mission
Range	xsd:string

has radius

Name	hasRadius
Definition	Indicates the radius around a location. It's used for representing areas and not points.
Instance of	owl:DatatypeProperty
Domain	Location
Range	xsd:int

has raw media source

Name	hasRawMediaSource
Definition	Links a media item to a URI pointing to the raw unprocessed item.
Instance of	owl:DatatypeProperty
Domain	MediaItem
Range	xsd:anyURI

has report ID

Name	hasReportID
Definition	Assigns and ID to an incident report.
Instance of	owl:DatatypeProperty
Domain	IncidentReport
Range	xsd:string

has unit

Name	hasUnit
Definition	Indicates the measurement unit of a climate parameter (e.g. Celsius for temperature measurements). Should be use together with property hasValue.
Instance of	owl:DatatypeProperty
Domain	ClimateParameter
Range	xsd:string

has value

Name	hasValue
Definition	Indicates the measurement value of a climate parameter. Should be used together with property hasUnit.
Instance of	owl:DatatypeProperty
Domain	ClimateParameter
Range	xsd:float

Appendix B – Competency Questions as SPARQL Queries

This appendix lists all the CQs presented in section 2.1 , along with their translation to SPARQL⁶ and the evaluation of the retrieved results, according to the process described in section 2.6.3 . As seen in the table, all of the CQs have been evaluated positively.

CQ#	Competency Question	Correct?
CQ1-1	<i>Which natural disasters may lead to natural disaster [X]?</i>	Yes
	<pre>SELECT ?disaster1 ?disaster2 WHERE { ?disaster1 rdf:type ba:NaturalDisasterType . ?disaster2 rdf:type ba:NaturalDisasterType . ?disaster1 ba:leadsTo ?disaster2 . }</pre>	
CQ1-2	<i>What are the impacts caused by natural disaster [X]?</i>	Yes
	<pre>SELECT ?impact ?disaster WHERE { ?impact rdf:type ba:ImpactType . ?disaster rdf:type ba:NaturalDisasterType . ?disaster ba:causesDisasterImpact ?impact . }</pre>	
CQ1-3	<i>Which climate parameters characterize natural disaster [X]?</i>	Yes
	<pre>SELECT ?parameter ?disaster WHERE { ?parameter rdf:type ba:ClimateParameterType . ?disaster rdf:type ba:NaturalDisasterType . ?disaster ba:characterizedByParameterType ?parameter . }</pre>	
CQ1-4	<i>What are the measurements for climate parameter [X] for natural disaster [Y]?</i>	Yes
	<pre>SELECT ?measurement ?disaster WHERE { ?measurement rdf:type ba:ClimateParameter . ?disaster rdf:type ba:NaturalDisaster . ?disaster ba:hasClimateParameterMeasurement ?measurement . }</pre>	
CQ1-5	<i>What is the average measurement for climate parameter [X] during natural disaster [Y]?</i>	Yes
	<pre>SELECT (AVG(?value) AS ?avg) WHERE { ?measurement rdf:type ba:ClimateParameter . ?disaster rdf:type ba:NaturalDisaster . ?disaster ba:hasClimateParameterMeasurement ?measurement . ?measurement ba:hasValue ?value . } GROUP BY ?disaster</pre>	

⁶ In these SPARQL queries, the "rdf" prefix indicates the namespace of the core W3C RDF vocabulary, while the "ba" prefix indicates the namespace of the beAWARE ontology.

CQ#	Competency Question	Correct?
CQ1-6	<i>Where did natural disaster [X] take place?</i>	Yes
	<pre> SELECT ?disaster ?location WHERE { ?disaster rdf:type ba:NaturalDisaster . ?location rdf:type ba:Location . ?disaster ba:hasDisasterLocation ?location . } </pre>	
CQ1-7	<i>What incidents are associated with natural disaster [X]?</i>	Yes
	<pre> SELECT ?disaster ?incident WHERE { ?disaster rdf:type ba:NaturalDisaster . ?incident rdf:type ba:Incident . ?disaster ba:hasRelatedIncident ?incident . } </pre>	
CQ1-8	<i>Where did incident [X], which is associated with natural disaster [Y], take place?</i>	Yes
	<pre> SELECT ?disaster ?incident ?location WHERE { ?disaster rdf:type ba:NaturalDisaster . ?incident rdf:type ba:Incident . ?location rdf:type ba:Location . ?disaster ba:hasRelatedIncident ?incident . ?incident ba:hasIncidentLocation ?location . } </pre>	
CQ1-9	<i>What are the impacts caused by incident [X] during natural disaster [Y]?</i>	Yes
	<pre> SELECT ?disaster ?incident ?impact WHERE { ?disaster rdf:type ba:NaturalDisaster . ?incident rdf:type ba:Incident . ?impact rdf:type ba:Impact . ?disaster ba:hasRelatedIncident ?incident . ?incident ba:hasIncidentImpact ?impact . } </pre>	
CQ1-10	<i>What is the location with the most incidents during natural disaster [X]?</i>	Yes
	<pre> SELECT ?location (COUNT(?incident) AS ?incidents) WHERE { ?disaster rdf:type ba:NaturalDisaster . ?incident rdf:type ba:Incident . ?location rdf:type ba:Location . ?disaster ba:hasRelatedIncident ?incident . ?incident ba:hasIncidentLocation ?location . } GROUP BY ?location ORDER BY DESC(?incidents) LIMIT 1 </pre>	
CQ1-11	<i>What incidents took place during 21-Jun-2017 during natural disaster [X]?</i>	Yes
	<pre> SELECT ?disaster ?incident ?start ?end WHERE { ?disaster rdf:type ba:NaturalDisaster . ?incident rdf:type ba:Incident . ?disaster ba:hasRelatedIncident ?incident . ?incident ba:hasIncidentStart ?start . ?incident ba:hasIncidentEnd ?end . FILTER ((?start >= "2017-06-21T00:00:00.000"^^xsd:dateTime) && (?end < "2017-06-22T00:00:00.000"^^xsd:dateTime)) } </pre>	

CQ#	Competency Question	Correct?
CQ1-12	<i>Which incidents during natural disaster [X] are the most severe?</i>	Yes
	<pre> SELECT ?disaster ?incident ?severity WHERE { ?disaster rdf:type ba:NaturalDisaster . ?incident rdf:type ba:Incident . ?disaster ba:hasRelatedIncident ?incident . ?incident ba:hasIncidentSeverity ?severity . FILTER (?severity = "high"^^xsd:string) } </pre>	
CQ1-13	<i>What is the priority of incident [X] during natural disaster [Y]?</i>	Yes
	<pre> SELECT ?disaster ?incident ?priority WHERE { ?disaster rdf:type ba:NaturalDisaster . ?incident rdf:type ba:Incident . ?disaster ba:hasRelatedIncident ?incident . ?incident ba:hasIncidentPriority ?priority . } </pre>	
CQ1-14	<i>What incidents during natural disaster [X] are the most urgent (i.e. with the highest priority)?</i>	Yes
	<pre> SELECT ?disaster ?incident ?priority WHERE { ?disaster rdf:type ba:NaturalDisaster . ?incident rdf:type ba:Incident . ?disaster ba:hasRelatedIncident ?incident . ?incident ba:hasIncidentPriority ?priority . FILTER (?priority = "high"^^xsd:string) } </pre>	
CQ2-1	<i>Where and when was media item [X] created?</i>	Yes
	<pre> SELECT ?item ?where ?when WHERE { ?item rdf:type ba:VideoItem . ?where rdf:type ba:Location . ?item ba:hasMediaLocation ?where . ?item ba:hasMediaItemTimestamp ?when . } </pre>	
CQ2-1	<i>Where and when was video [X] created?</i>	Yes
	<pre> SELECT ?item ?where ?when WHERE { ?item rdf:type ba:VideoItem . ?where rdf:type ba:Location . ?item ba:hasMediaLocation ?where . ?item ba:hasMediaItemTimestamp ?when . } </pre>	
CQ2-2	<i>What is the location with the most videos created?</i>	Yes
	<pre> SELECT ?location (COUNT(?item) AS ?items) WHERE { ?location rdf:type ba:Location . ?item rdf:type ba:VideoItem . ?item ba:hasMediaLocation ?location . } GROUP BY ?location ORDER BY DESC(?items) LIMIT 1 </pre>	

CQ#	Competency Question	Correct?
CQ2-3	<i>Which vulnerable objects were involved in incident [X]?</i>	Yes
	<pre> SELECT ?incident ?object WHERE { ?incident rdf:type ba:Incident . ?incident ba:involvesParticipant ?object . } </pre>	
CQ2-4	<i>What impact do the vulnerable objects involved in incident [X] suffer?</i>	Yes
	<pre> SELECT ?incident ?object ?impact WHERE { ?incident rdf:type ba:Incident . ?object rdf:type ba:VulnerableObject . ?incident ba:involvesParticipant ?object . ?object ba:isImpactedBy ?impact . } </pre>	
CQ2-5	<i>What is the risk suffered by vulnerable objects involved in incident [X]?</i>	Yes
	<pre> SELECT ?incident ?object ?risk WHERE { ?incident rdf:type ba:Incident . ?object rdf:type ba:VulnerableObject . ?detection rdf:type ba:Detection . ?incident ba:involvesParticipant ?object . ?detection ba:detectsParticipant ?object . ?detection ba:hasDetectionRisk ?risk . } </pre>	
CQ2-6	<i>What are the vulnerable objects that suffer the greatest risk during incident [X]?</i>	Yes
	<pre> SELECT ?incident ?object ?risk WHERE { ?incident rdf:type ba:Incident . ?object rdf:type ba:VulnerableObject . ?detection rdf:type ba:Detection . ?incident ba:involvesParticipant ?object . ?detection ba:detectsParticipant ?object . ?detection ba:hasDetectionRisk ?risk . } ORDER BY DESC(?risk) LIMIT 1 </pre>	
CQ2-7	<i>What is the detection confidence level for vulnerable object [X] during incident [Y]?</i>	Yes
	<pre> SELECT ?incident ?object ?confidence WHERE { ?incident rdf:type ba:Incident . ?object rdf:type ba:VulnerableObject . ?detection rdf:type ba:Detection . ?incident ba:involvesParticipant ?object . ?detection ba:detectsParticipant ?object . ?detection ba:hasDetectionConfidence ?confidence . } </pre>	

CQ#	Competency Question	Correct?
CQ2-8	<i>What are the vulnerable objects with the lowest confidence level detected during incident [X]?</i>	Yes
	<pre> SELECT ?incident ?object ?confidence WHERE { ?incident rdf:type ba:Incident . ?object rdf:type ba:VulnerableObject . ?detection rdf:type ba:Detection . ?incident ba:involvesParticipant ?object . ?detection ba:detectsParticipant ?object . ?detection ba:hasDetectionConfidence ?confidence . } ORDER BY ASC(?confidence) LIMIT 1 </pre>	
CQ2-9	<i>Which media items led to the creation of incident [X]?</i>	Yes
	<pre> SELECT ?incident ?item WHERE { ?incident rdf:type ba:Incident . ?item rdf:type ba:MediaItem . ?dataset rdf:type ba:Dataset . ?task rdf:type ba:Task . ?dataset ba:involvesIncident ?incident . ?task ba:taskProducesDataset ?dataset . ?task ba:relatesToMediaItem ?item . } </pre>	
CQ3-1	<i>What is the location of rescue unit [X]?</i>	Yes
	<pre> SELECT ?unit ?location WHERE { ?unit rdf:type ba:Responder . ?location rdf:type ba:Location . ?unit ba:hasResponderLocation ?location . } </pre>	
CQ3-2	<i>What is the mission assigned to rescue unit [X] and what is its current status?</i>	Yes
	<pre> SELECT ?unit ?mission ?status WHERE { ?unit rdf:type ba:Responder . ?mission rdf:type ba:Mission . ?unit ba:isAssignedMission ?mission . ?mission ba:hasMissionStatus ?status . } </pre>	
CQ3-3	<i>What is the location where rescue mission [X] is taking place?</i>	Yes
	<pre> SELECT ?mission ?location WHERE { ?mission rdf:type ba:Mission . ?location rdf:type ba:Location . ?unit rdf:type ba:Responder . ?unit ba:hasResponderLocation ?location . ?unit ba:isAssignedMission ?mission . } </pre>	

CQ#	Competency Question	Correct?
CQ3-4	<i>What is the incident that rescue unit [X] is addressing?</i>	Yes
	<pre> SELECT ?unit ?mission ?incident WHERE { ?unit rdf:type ba:Responder . ?mission rdf:type ba:Mission . ?incident rdf:type ba:Incident . ?unit ba:isAssignedMission ?mission . ?mission ba:relatesToIncident ?incident . } </pre>	
CQ3-5	<i>What are the vulnerable objects involved in the mission assigned to rescue unit [X]?</i>	Yes
	<pre> SELECT ?unit ?mission ?object WHERE { ?unit rdf:type ba:Responder . ?mission rdf:type ba:Mission . ?object rdf:type ba:VulnerableObject . ?incident rdf:type ba:Incident . ?unit ba:isAssignedMission ?mission . ?mission ba:relatesToIncident ?incident . ?incident ba:involvesParticipant ?object . } </pre>	
CQ3-6	<i>What is the potential impact of the mission assigned to rescue unit [X]?</i>	Yes
	<pre> SELECT ?unit ?mission ?impact WHERE { ?unit rdf:type ba:Responder . ?mission rdf:type ba:Mission . ?impact rdf:type ba:Impact . ?unit ba:isAssignedMission ?mission . ?mission ba:relatesToIncident ?incident . ?incident ba:hasIncidentImpact ?impact . } </pre>	
CQ3-7	<i>What rescue missions have taken place during 21-Jun-2017?</i>	Yes
	<pre> SELECT ?mission WHERE { ?mission rdf:type ba:Mission . ?mission ba:hasMissionStart ?start . ?mission ba:hasMissionEnd ?end . FILTER ((?start >= "2017-06-21T00:00:00.000"^^xsd:dateTime) && (?end <= "2017-06-22T00:00:00.000"^^xsd:dateTime)) } </pre>	
CQ3-8	<i>Where is the most urgent mission (i.e. the one with the highest priority) taking place?</i>	Yes
	<pre> SELECT ?mission ?unit ?priority WHERE { ?mission rdf:type ba:Mission . ?unit rdf:type ba:Responder . ?unit ba:isAssignedMission ?mission . ?mission ba:hasMissionPriority ?priority . FILTER (?priority = "high"^^xsd:string) } </pre>	

CQ#	Competency Question	Correct?
CQ3-9	<i>Which rescue unit is assigned the most severe incident?</i>	Yes
	<pre>SELECT ?unit ?mission ?incident ?severity WHERE { ?unit rdf:type ba:Responder . ?mission rdf:type ba:Mission . ?incident rdf:type ba:Incident . ?unit ba:isAssignedMission ?mission . ?mission ba:relatesToIncident ?incident . ?incident ba:hasIncidentSeverity ?severity . FILTER (?severity = "high"^^xsd:string) }</pre>	