Testbed-12 General Feature Model Engineering Report

Table of Contents

1. Introduction 6
   1.1. Scope 6
   1.2. Document contributor contact points 7
   1.3. Future Work 7
   1.4. Foreword 8
2. References 9
3. Terms and definitions 10
   3.1. Abbreviated terms 10
4. Overview 11
5. Initial Requirements, Assumptions and Concepts 12
   5.1. Integration of Data, Analytical Observations and Judgments 12
   5.2. Correlation does not necessarily mean causation 13
   5.3. Making Judgments on Facts 13
   5.4. Interaction with an Integrated Information Environment 13
      5.4.1. Activity Based Intelligence (ABI) 14
      5.4.2. Object Based Production (OBP) 14
      5.4.3. Structured Observation Management (SOM) 14
   5.5. Relationships - Features and Associations 14
   5.6. Use Case Scenario 14
   5.7. Available Scenario Phases 15
   5.8. Scenario Script 16
   5.9. Accommodating Associations in GFM 17
   5.10. WOS Approaches 19
      5.10.1. CSW ebRIM based WOS Approach 20
      5.10.2. WFS based WOS Approach 21
   5.11. Client Requirements 21
   5.12. Initial Implementation Experiences 22
      5.12.1. Data Availability 22
      5.12.2. Data Georeferencing 22
6. The General Feature Model 23
   6.1. Geographic Information: Key Principles 23
   6.2. Metamodel for Geographic Information: The General Feature Model 23
7. CSW-eb-RIM based WOS Implementation 27
   7.1. Interaction Model 28
   7.2. Searching for WOS Objects 28
   7.3. Interrogation of a WOS Object 28
   7.4. Object Insertion 29
   7.5. Update and Deletion 29

8. WFS based Web Object Service 30
8.1. Introduction 30
8.2. Requirements 30
8.3. Architecture 31
8.4. Basic Service Elements 31
8.4.1. Service Root 31
8.4.2. Object Representation 31
8.4.3. Content Negotiation 32
8.4.4. Exceptions 32
8.5. Authentication and Access Control 32
8.6. Hypermedia Controls 33
8.6.1. Introduction 33
8.6.2. Hypermedia Control in a WOS' Capabilities Document 33
8.6.3. Hypermedia Controls in a WOS' Query Response 33
8.7. Summary of Resources 33
8.8. MIME Types 34
8.9. A Word about Examples 34
8.10. Service Metadata 35
8.10.1. Introduction 35
8.10.2. Resources 35
8.10.3. Query Parameters 35
8.10.4. Representations 35
8.10.5. Methods 36
8.10.6. Examples 36
8.11. Object Access & Management (GetObject/Transaction) 41
8.11.1. Introduction 41
8.11.2. Resources 41
8.11.3. Query parameters 42
8.11.4. Representations 42
8.11.5. Methods 43
8.11.6. Examples: 44
8.12. Associations 47
8.12.1. Introduction 47
8.12.2. Resources 47
8.12.3. Methods 49
8.12.4. Examples 50
8.13. Classifications 52
8.13.1. Introduction 52
8.13.2. Resources 53
8.13.3. Methods 56
8.13.4. Examples 58

9. Future Work 61
Appendix A: Revision History 63
Appendix B: Bibliography 64

Publication Date: 2017-05-12

Approval Date: 2016-12-07

Posted Date: 2016-11-21

Reference number of this document: OGC 16-047r1

Reference URL for this document: http://www.opengis.net/doc/PER/t12-A075

Category: Public Engineering Report

Editor: Martin Klopfer

Title: Testbed-12 General Feature Model Engineering Report

Testbed-12 General Feature Model Engineering Report (16-047r1)

COPYRIGHT

WARNING

This document is an OGC Public Engineering Report created as a deliverable of an initiative from the OGC Innovation Program (formerly OGC Interoperability Program). It is not an OGC standard and not an official position of the OGC membership.It is distributed for review and comment. It is subject to change without notice and may not be referred to as an OGC Standard. Further, any OGC Engineering Report should not be referenced as required or mandatory technology in procurements. However, the discussions in this document could very well lead to the definition of an OGC Standard.

LICENSE AGREEMENT

Permission is hereby granted by the Open Geospatial Consortium, ("Licensor"), free of charge and subject to the terms set forth below, to any person obtaining a copy of this Intellectual Property and any associated documentation, to deal in the Intellectual Property without restriction (except as set forth below), including without limitation the rights to implement, use, copy, modify, merge, publish, distribute, and/or sublicense copies of the Intellectual Property, and to permit persons to whom the Intellectual Property is furnished to do so, provided that all copyright notices on the intellectual property are retained intact and that each person to whom the Intellectual Property is furnished agrees to the terms of this Agreement.

If you modify the Intellectual Property, all copies of the modified Intellectual Property must include, in addition to the above copyright notice, a notice that the Intellectual Property includes modifications that have not been approved or adopted by LICENSOR.

THIS LICENSE IS A COPYRIGHT LICENSE ONLY, AND DOES NOT CONVEY ANY RIGHTS UNDER ANY PATENTS THAT MAY BE IN FORCE ANYWHERE IN THE WORLD. THE INTELLECTUAL PROPERTY IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE DO NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE INTELLECTUAL PROPERTY WILL MEET YOUR REQUIREMENTS OR THAT THE OPERATION OF THE INTELLECTUAL PROPERTY WILL BE UNINTERRUPTED OR ERROR FREE. ANY USE OF THE INTELLECTUAL PROPERTY SHALL BE MADE ENTIRELY AT THE USER’S OWN RISK. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR ANY CONTRIBUTOR OF INTELLECTUAL PROPERTY RIGHTS TO THE INTELLECTUAL PROPERTY BE LIABLE FOR ANY CLAIM, OR ANY DIRECT, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM ANY ALLEGED INFRINGEMENT OR ANY LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR UNDER ANY OTHER LEGAL THEORY, ARISING OUT OF OR IN CONNECTION WITH THE IMPLEMENTATION, USE, COMMERCIALIZATION OR PERFORMANCE OF THIS INTELLECTUAL PROPERTY.

This license is effective until terminated. You may terminate it at any time by

destroying the Intellectual Property together with all copies in any form. The license will also terminate if you fail to comply with any term or condition of this Agreement. Except as provided in the following sentence, no such termination of this license shall require the termination of any third party end-user sublicense to the Intellectual Property which is in force as of the date of notice of such termination. In addition, should the Intellectual Property, or the operation of the Intellectual Property, infringe, or in LICENSOR's sole opinion be likely to infringe, any patent, copyright, trademark or other right of a third party, you agree that LICENSOR, in its sole discretion, may terminate this license without any compensation or liability to you, your licensees or any other party. You agree upon termination of any kind to destroy or cause to be destroyed the Intellectual Property together with all copies in any form, whether held by you or by any third party.

Except as contained in this notice, the name of LICENSOR or of any other holder of a copyright in all or part of the Intellectual Property shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Intellectual Property without prior written authorization of LICENSOR or such copyright holder. LICENSOR is and shall at all times be the sole entity that may authorize you or any third party to use certification marks, trademarks or other special designations to indicate compliance with any LICENSOR standards or specifications.

This Agreement is governed by the laws of the Commonwealth of Massachusetts. The application to this Agreement of the United Nations Convention on Contracts for the International Sale of Goods is hereby expressly excluded. In the event any provision of this Agreement shall be deemed unenforceable, void or invalid, such provision shall be modified so as to make it valid and enforceable, and as so modified the entire Agreement shall remain in full force and effect. No decision, action or inaction by LICENSOR shall be construed to be a waiver of any rights or remedies available to it.

None of the Intellectual Property or underlying information or technology may be downloaded or otherwise exported or reexported in violation of U.S. export laws and regulations. In addition, you are responsible for complying with any local laws in your jurisdiction which may impact your right to import, export or use the Intellectual Property, and you represent that you have complied with any regulations or registration procedures required by applicable law to make this license enforceable.

Abstract

With a growing requirement to carry out complex analysis in large multi-disciplinary, heterogeneous data collections, an approach is required to extract equivalent information from dissimilar content. The more information can be normalized, the easier it will be to correlate the content. Given that almost all data has a spatio-temporal component, this ER will look into the idea of defining a Spatial-Temporal Service and analyze which collection of data types, operations and architecture patterns would be necessary to spatial-temporal enable any content. This OGC® document reviews the General Feature Model and gives guidelines for necessary modifications to broaden its scope, so that it can be re-used for non-geospatial centric applications and extended as necessary into a general model for all object types.

Business Value

The GFM and it's capabilities have long been regarded as being of interest to the OGC / the geospatial community only. However, the proof-of-concept carried out in this activity has successfully demonstrated, that the GFM can accommodate far more requirements than anticipated. Even rather abstract items such as a judgement, a decision or the related body of evidence can be expressed as features and associations. Given that almost all information can be attributed with some spatio-temporal tag, a GFM based approach can play an important role in the integration of geospatial and non-geospatial data and systems.

Technology Value

This ER summarizes the work performed in Testbed-12 and provides an outlook on possible future activities. It serves as a starting point for the OGC community in general and the Geosemantics Working Group in particular to understand some of the latest discussions on semantics in geospatial contexts. It provides a number of references to more detailed material to facilitate more in-depth research and analysis.

Although this ER does not reflect the work of any OGC Working Group, it addresses topics which are currently discussed in the Defense & Intelligence DWG, Disaster Management DWG and Big Data DWG. The initial GEOINT requirements were derived from capabilities anticipated for Activity Based Intelligence (ABI), Object Based Production (OBP) and Structured Observation Management (SOM). It concentrates on summarizing the activities performed in Testbed-12 that were carried out to analyze the capabilities of the GFM to store non-geospatial data and associations. As such it provides a starting point for

future work on geospatial and non-geospatial data integration.

Keywords

ogcdocs, testbed-12, General Feature Model, heterogeneous data integration

Proposed OGC Working Group for Review and Approval

WFS/FES SWG and Defence and Intelligence DWG

Chapter 1. Introduction

1.1. Scope

Given that almost all data has a spatial-temporal component, this OGC® Engineering Report evaluates the degree to which the General Feature Model can accommodate complex analysis on large multi-disciplinary, heterogeneous data collections for the purpose of extracting new information from dissimilar content. It looks into the idea of defining a Spatial-Temporal Service and Analytics platform and identifies the data types, operations and architecture patterns would be necessary to spatial-temporal enable any type of content.

This OGC® document reviews the General Feature Model and gives guidelines for necessary modifications to broaden its scope, so that it can be re-used for non-geospatial centric applications and extended as necessary into a general model for all object types.

The idea to integrate information and capture knowledge has evolved in OGC over a long period of time, accommodating the continuous changes in technology and the range of services to be integrated. The following OGC documents summarize some of this work and have been considered in the development of the initial concepts:

The Web Object Service Implementation Specification (OGC 02-049 / 03-013) aimed to satisfy a requirement of the OWS 1.2 project to manage many different types of objects, including styles, symbols and images, using a repository interface. “The specification defines a set of base XML types that define the behavior of a Web Object Service. A Web Object Service is a generic web-based repository interface. The interface supports the following operations: GetCapabilities, DescribeObjectType, GetObjectById, GetObject, LockObject and Transaction. The specification assumes that the distributed computing platform is HTTP and may define both XML (suitable for the POST method) and Keyword-Value Pair (suitable for the GET method) encodings of each operation.”

The OGC® Testbed-10 Service Integration Engineering Report (OGC 14-13r1) “specifies a means of discovering and describing associations between web resources (both OGC and non-OGC). The discovery of associations is accomplished by means a new operation named GetAssociations. The description or representation of associations is accomplished by the definition of an XML Schema for expressing associations in XML. The scope of this document is limited to the existing OGC web service framework, which predominantly uses XML. However, it is anticipated that over time the association schemas defined in this document will be extended to provide JSON snd JSON-LD expressions which are rapidly emerging encodings within and outside the OGC.”

The OGC® Testbed-11 Incorporating Social Media in Emergency Response Engineering Report (OGC 15-057) studies two different approaches to incorporating social media information into Emergency Response applications: “The first [approach] was based on the concept of using an OGC Sensor Observation Service (SOS) for handling humans as sensors is illustrated from an interoperability perspective. The developed O&M-based data model is presented and illustrated with examples from different social media platforms. This is complemented by a description of how data loading from social media platforms into an SOS server can be achieved. The second approach was based on using Linked Data to integrate social objects produced by the different social media networks. The ontology for describing social objects and activities is based on SocialML ontologies that can be

extended to accommodate new activities and social objects. The integration of social sites was done through the use of RDF scrapers accessible through a REST API. The resulting knowledgebase was made available through a GeoSPARQL endpoint. Another important topic of [OGC 15-057] is the use of the OGC Web Processing Service (WPS) for analyzing the available social media data (i.e. detect clusters). “

However, the requirement to integrate heterogenous data using spatial-temporal attributes has also been addressed outside the OGC. One example explicitly mentioned in the Testbed-12 RFQ is the FGDC endorsed Time Space Position Information (TSPI) Version 2.0 standard (NGA.STND.0019_2.0), which "provides a single means of encoding spatiotemporal information for the storage, manipulation, interchange, and exploitation of spatiotemporal data. It is designed as a set of reusable data components upon which both Geography Markup Language (GML)-based application schemas and non-GML-based XML schemas may be developed. It specifies a registry-based extension mechanism enabling the development and reuse of additional spatiotemporal XML Schema components. It integrates the XML Schema for the United States Thoroughfare, Landmark, and Postal Address Data Standard, FGDC-STD-016-2011, including the necessary updates to work with the OGC® GML 3.3, Extended schemas and encoding rules."

The Testbed-12 General Feature Model ER now takes a step back from specific implementation requirements and looks into the principle capabilities of the GFM to accommodate non-geospatial data, information and associations as features.

A simple use case scenario is based on observations from non-geospatial sources, subsequent decisions based on selected observations and a requirement to create associations, which allow a user to access all associated features. The objective is to make a body of evidence for a decision accessible from various angles. This can be the traditional geospatial approach of showing a decision on a map and providing a report of associated features. However it should also be capable of providing graph-like overviews of features and associations accessible from non-geospatial systems.

1.2. Document contributor contact points

All questions regarding this document should be directed to the editor or the contributors:

Table 1. Contacts

Name	**Organization**
Roger Brackin	Envitia
Charles Heazel	WiSC Enterprises
Gobe Hobona PhD.	Envitia
Martin Klopfer	Frisia IT
Panagiotis (Peter) A. Vretanos	CubeWerx Inc.

1.3. Future Work

This initiative has demonstrated the feasibility of a Spatial-Temporal Service and Analytics Platform

based on the General Feature Model. However, much work remains to be done to fully define and validate that platform. The proposed next steps in this effort can be found in Section 9.

The work carried out in this activity has not resulted in any change requests towards existing specifications.

1.4. Foreword

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

Chapter 2. References

The following documents are referenced in this document. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. For undated references, the latest edition of the normative document referred to applies.

- OGC OGC 06-121r9, OWS Common Implementation Standard
- OGC 16-046, Testbed-12 Semantic Enablement ER
- OGC 02-049 / 03-013, Web Object Service Implementation Specification
- OGC 14-13r1, Testbed-10 Service Integration Engineering Report
- OGC 15-057, Testbed-11 Incorporating Social Media in Emergency Response Engineering Report
- NGA.STND.0019_2.0, Time Space Position Information (TSPI) Version 2.0 standard
- NGA, Everything you wanted to know about the NSG Application Schema (NAS), TB12 Reference Material Presentation

Chapter 3. Terms and definitions

For the purposes of this report, the definitions specified in Clause 4 of the OWS Common Implementation Standard [OGC 06-121r9] shall apply. In addition, the following terms and definitions apply.

3.1. Abbreviated terms

- ABI Activity Based Intelligence
- API Application Program Interface
- COM Component Object Model
- CSW Catalogue Service for the Web
- DCE Distributed Computing Environment
- DCOM Distributed Component Object Model
- ebRIMElectronic Business Registry Information Model
- ER Engineering Report
- GFM General Feature Model
- GMLSF Simple Feature Geography Markup Language
- IDL Interface Definition Language
- LDP Linked Data Platform
- NGA National Geospatial-Intelligence Agency
- OBP Object Based Production
- OWL Web Ontology Language
- OWL-S Semantic Markup for Web Services
- RDF Resources Description Format
- RDF-QB RDF Data Cube
- SHACL Shapes Constraint Language
- SKOS Simple Knowledge Organization System
- SOM Structured Object Management
- SPARQL SPARQL Protocol and RDF Query Language
- UML Unified Modeling Language
- VoID Vocabulary of Interlinked Datasets
- WOS Web Object Service

Chapter 4. Overview

This ER serves as an entry point to Testbed-12 activities looking into the idea of using the GFM as a basis for integration of heterogeneous and multi-temporal geospatial and non-geospatial data. It integrates discussions from the *Linked Data and Advanced Semantics for Data Discovery and Dynamic Integration* thread.

The main document starts in Clause 5 with a short overview of the initial requirements, assumptions and concepts, as well as the use case scenario selected for a proof of concept demonstration.

It then maps the concepts to the use case scenario and derives an initial model based on the GFM in Clause 6.

Two demo implementations carried out by Envitia and Cubewerx are briefly described and discussed in Clause 7 and Clause 8.

The experiences from the proof of concept implementations as well as future work items are finally summarized inClause 9.

Chapter 5. Initial Requirements, Assumptions and Concepts

With a growing requirement to carry out complex analysis in large multi-disciplinary, heterogeneous data collections, an approach is required to extract equivalent information from dissimilar content. The more information can be normalized, the easier it will be to correlate the content.

Given that almost all data has a spatio-temporal component, this ER looks into the idea of defining a Spatial-Temporal Platform as a Web Object Service (WOS) and analyzes which collection of data types, operations and architecture patterns would be necessary to spatial-temporal enable any content.

It is assumed that the General Feature Model, WFS and CSW already accommodate most capabilities, although by nature of their origin, they are primarily geared towards geospatial application.

The envisioned WOS is not about merging WFS and CSW. It is about being able to serve any type of object, e.g. vector features, raster coverages, other images, sounds, or even representations of a concept or conclusion.

The WOS is thus a means to handle associations as a first class object to allow the description of relationships between arbitrary objects in a formal way as described below.

The goal is to review the General Feature Model and to research necessary modifications to broaden its scope, so that it can be re-used for non-geospatial centric applications and extended as necessary into a general model for all object types.

5.1. Integration of Data, Analytical Observations and Judgments

As part of their homeland security mission, NGA has compiled the HSIP Gold database. HSIP Gold is a compilation of 560 of the best available geospatially enabled baseline infrastructure datasets for all National & Defense Critical Infrastructure Sectors. Now they want to extend that capability by providing not just feature data, but a comprehensive integrated information environment. This environment provides a comprehensive suite of integrated information as well as analytics to derive knowledge from that information.

Central to this capability is a data aggregation service: Data is collected from open source, crowd sources, and procured from key data providers.

All data is assigned a unique identifier, and populated with discovery metadata. That metadata includes the spatial and temporal extent. Core to the data store is a global feature data set. This foundation feature data is a detailed representation of the global real-world objects. The foundation feature data provides a spatial-temporal context against which all other data can be understood.

The data holdings do not stop with data. They also include the results of analytic processes applied

to the data. These results are classified as observations or judgments.

5.2. Correlation does not necessarily mean causation

Observations identify correlations between data. These may be generated by observation or through automated analytics. Correlations are statements of fact, but do not indicate meaning.

For example, two suspected criminal operatives are in San Francisco at the same time and stayed in hotels within ½ mile from each other. This could indicate that they are planning an attack. It could be simple coincidence. The observation is a simple observance of fact. It makes no judgment as to the significance of that fact.

5.3. Making Judgments on Facts

Judgments are the result of analytic processes, which develop conclusions from a body of evidence. They can be the result of automated or human analytic processes. A judgment must include:

- the entity which generated the judgment,
- the analytic process used,
- the body of evidence supporting the judgment, which can include data, observations and other judgments,
- a measure of confidence for the judgment, which should reflect the aggregate confidence in each evidence entity, as well as the confidence in the analytic process. A change in any of the evidence should result in a re-calculation of, or at last a flag against, the confidence in the judgment.

This information is sufficient for a user to review the analytic process and make their own assessment as to its value.

5.4. Interaction with an Integrated Information Environment

Data, observations and judgments reside in the integrated information environment and all information is spatial-temporally enabled.

This information can be compiled for distribution through a report, i.e. a compilation of observations and judgments intended for human consumption. However, such reports do not have to be in a document format. A report may also be an interactive web site, which allows users to visualize and navigate through the judgments and reasoning. They may also be dynamically updated as the underlying information changes.

Users can build their own analytics or use the analytics provided by the service. Analytics can be assembled into workflows resulting in more complex analytics.

This concept is also well described in NGA’s presentation ""Everything you wanted to know about the NSG Application Schema (NAS)"". The NAS document describes three major components of this

integrated information environment:

5.4.1. Activity Based Intelligence (ABI)

An analytic method applied to structured data from all sources, to discover objects, relationships, or behaviors by resolving significant activity. ABI methods accelerate and deepen insight and the ability to do OBP; new insight enriches models and understanding of adversary behaviors and relationships.

5.4.2. Object Based Production (OBP)

A framework for organizing and sharing information, relating observations from all sources to known objects (be they units, people, locations, or events).

5.4.3. Structured Observation Management (SOM)

NGA's framework for organizing and sharing GEOINT information, normalizing how to capture and record observations from all sensors and sources. SOM enables integration of GEOINT in OBP.

5.5. Relationships - Features and Associations

Based on the above mentioned conceptual ideas a feature can be almost anything. To illustrate the concept, the chosen data for the Use Case Scenario described below includes analytical observations, judgements, tweets, images and even a police radio message.

The crucial step is to move from individual features to linked features. The association describes the relationship between features and is thus the binding connection, linking objects with each other. This GFM approach works with every type of relationship.

The topic of associations has also come up in other TB12 activities and is further described in TB12 - Evaluate Semantic Enablement ER.

5.6. Use Case Scenario

To evaluate the degree to which the GFM is capable of capturing information from social media and related human judgments and decisions as features, a simple use case scenario has been developed:

- A large crowd moves from A to B along a predefined route,
- an incident, which could temporarily block the anticipated route is reported,
- the situation needs to be judged and
- decisions need to be taken and potentially justified later on.

Apart from the fabricated incident at the junction, this is based on a Climate Change Protest March with 5,000+ participants. The event took place in Oakland on 7th February 2015. It was well covered by tweets before, during and after the event via @DontFrackCA.

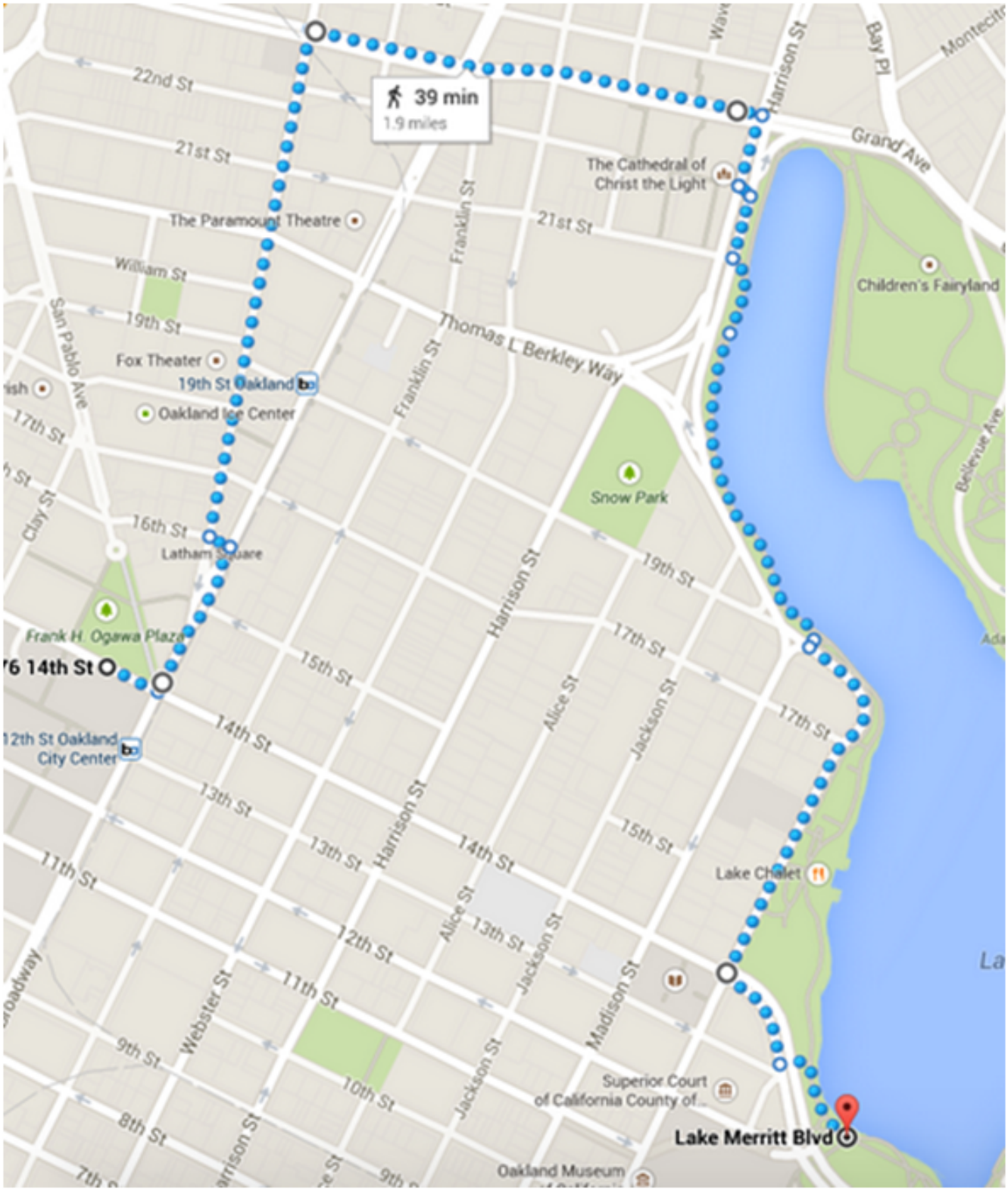

Figure 1. Planned route from Oakland City Hall, Frank H. Ogawa Plaza, intending to march to Lakeside Park

5.7. Available Scenario Phases

The use case scenario and it's associated data cover all three phases of a planned event:

During the pre-event phase, information can be derived about the topic, location and date, as well as the anticipated size and nature of the event. In the case of a planned event, where organizers and local authorities work together, this might provide information to judge the degree to which

planned and actual developments align. In case of an ad hoc event, it could provide early warning indicators to trigger actions for preparedness.

During the actual event phase, social media information can be used to support situation awareness and handling. For TB12 we have inserted a tweet message reporting a suspicious package next to a junction the Climate Change Protest March plans to pass on the planned route. This requires organizers to make decisions how to proceed. The decisions are based on the current location of the crowd, which could be machine-processed information from tweets, human observations reported by radio and judgments on available information. Observations, judgments and decisions shall all be treated a features with a traceable relationship to each other.

Post-event information might be relevant to rate the effectiveness of decisions resulting actions.

5.8. Scenario Script

Given that the scope of the GFM activity in TB12 is to proof the concept of storing heterogeneous observations and their associations in a GFM based model, it was decided to break down the scenario to a very simple chain of events:

- A suspicious parcel is observed on a junction of the planned route and triggers a potential threat alert
- A decision to investigate is taken based on the location of the crowd relative to the reported location
- The parcel can be identified and the threat alert is cleared

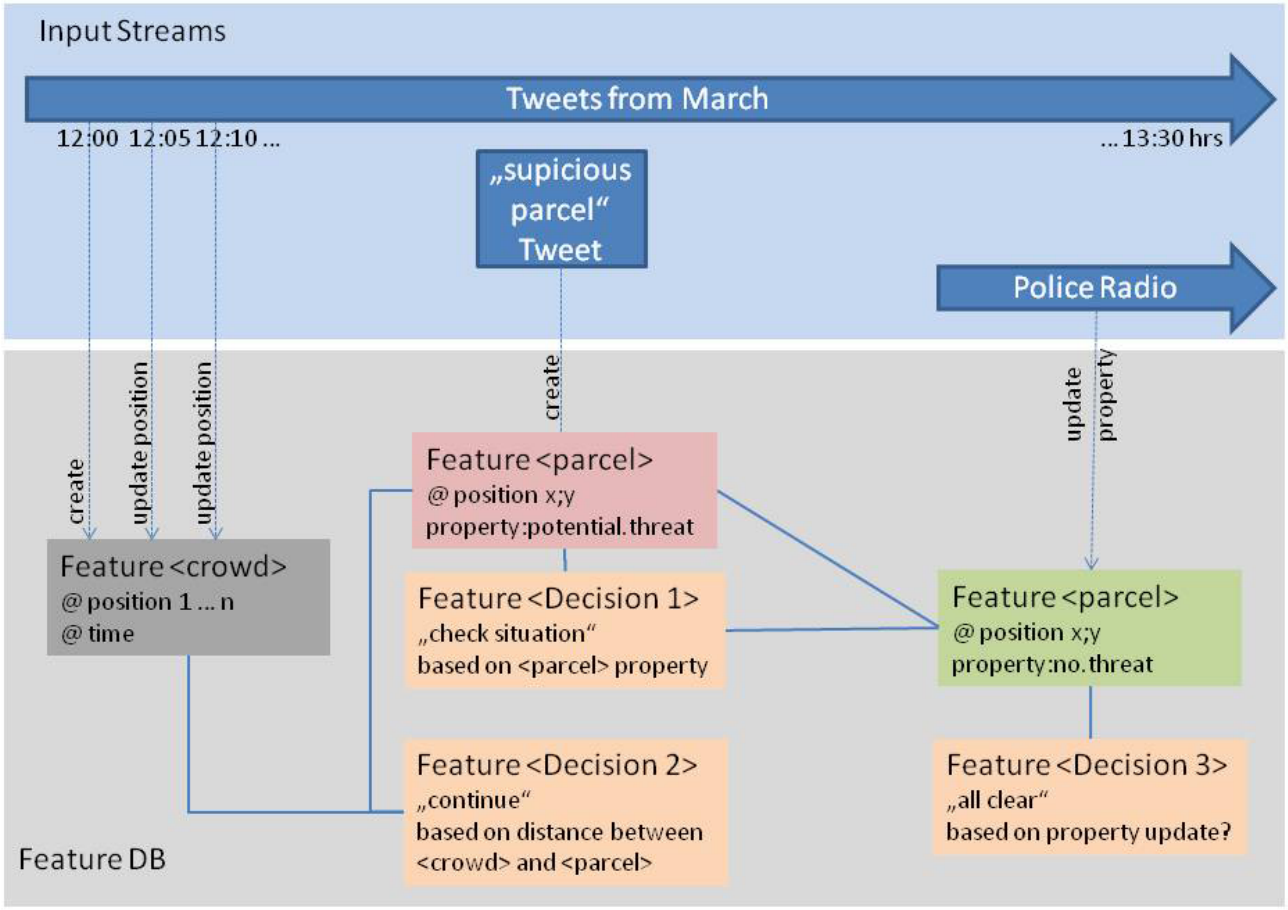

Figure 2. Conceptional View of Selected Scenario

5.9. Accommodating Associations in GFM

One of the key requirements in the TB12 GFM activity is not only to analyze to what degree non-geospatial objects can be accommodated in the GFM, but also to describe associations between them in order to maintain traceability, e.g. between an observation and a subsequent decision.

The GFM defines two relationships between GF_FeatureType and GF_AssociationType as illustrated below:

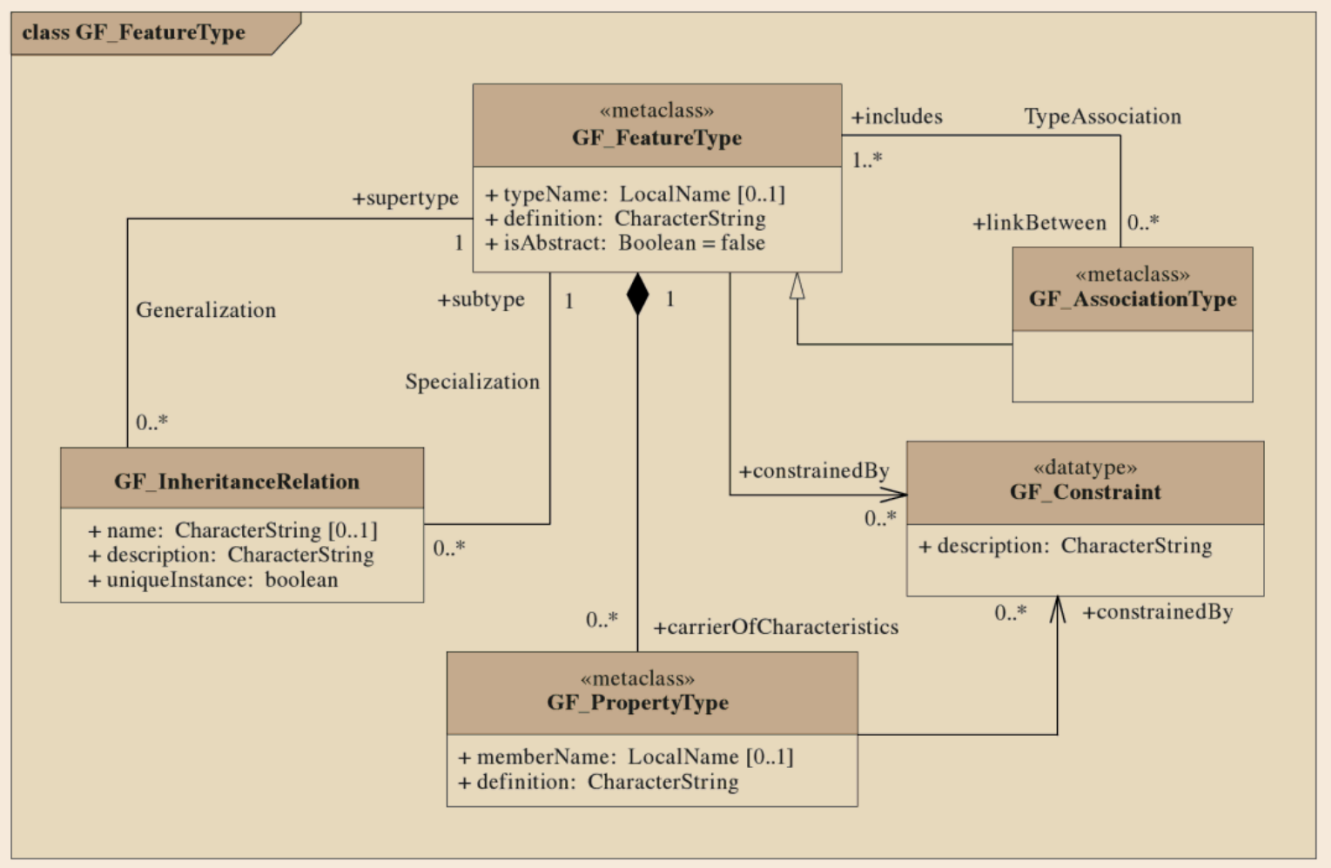

Figure 3. relationships between GF_FeatureType and GF_AssociationType

The "TypeAssociation" between GF_FeatureType and GF_AssociationType indicates that a GF_AssociationType instance can serve as the "linkBetween" two GF_FeatureType instances. That is, the GF_Association "includes" the associated GF_Feature instances. Since the GF_AssociationType is a metaclass derived from GF_FeatureType, this approach allows associations to have properties (attributes, associations, and operations) analogous to any other instance of GF_FeatureType.

The second relationship between GF_FeatureType and GF_AssociationType is an inheritance relationship. GF_AssociationType inherits all associations and properties of GF_FeatureType, its superclass. Since GF_AssociationType is a subclass of GF_FeatureType, then GF_Associations are also GF_Features. If GF_Associations are also GF_Features, then there can be GF_Association instances which associate GF_Associations. That is, associations between associations. This leap allows us to not only establish smart associations between objects, but to build a body of evidence (a collection of objects and associations) which asserts and supports the confidence level for each association.

In our example, we use this model to define additional properties for associations. The BodyOfEvidence Association class allows us to define the trust level and a reason, as illustrated below:

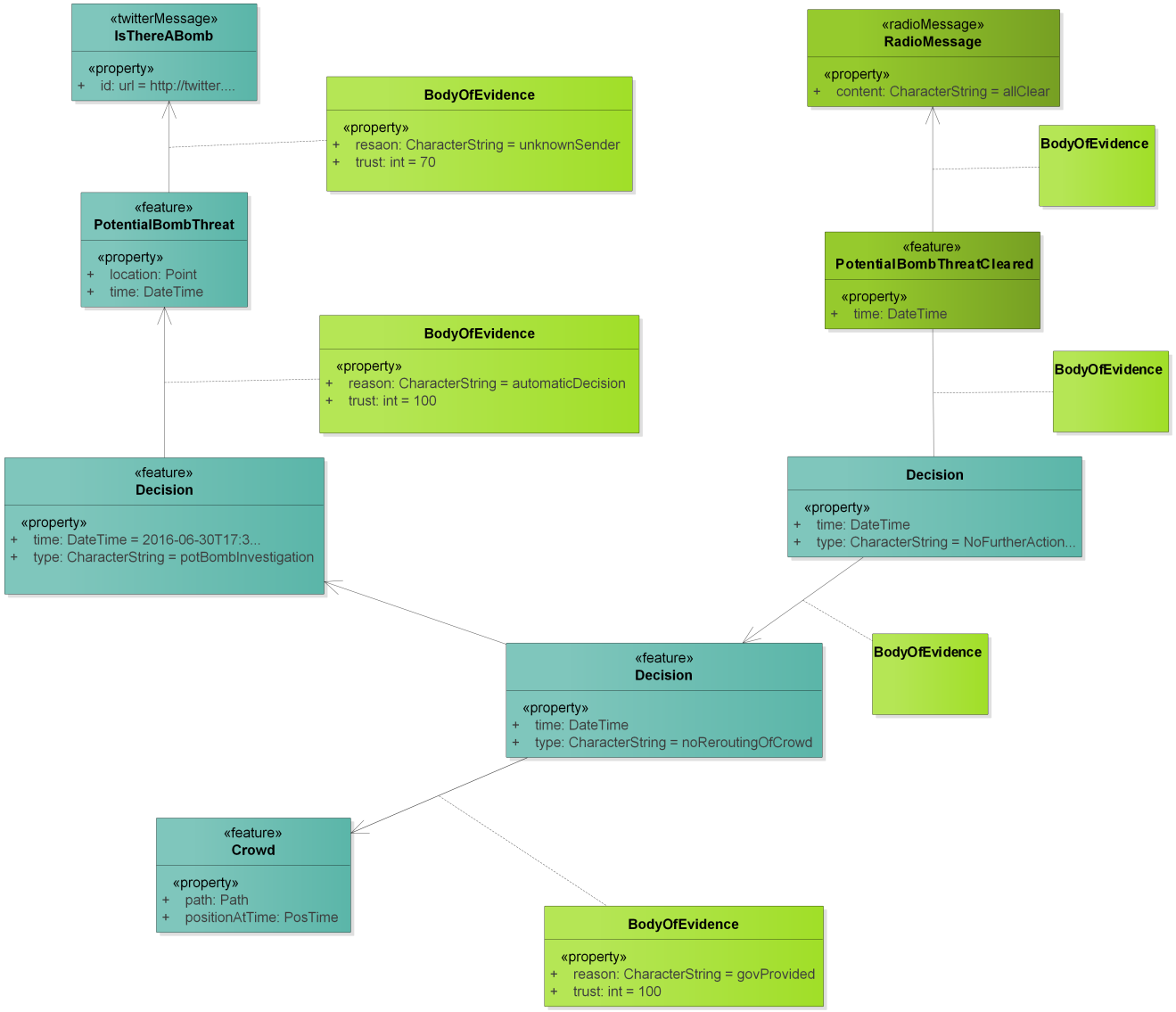

Figure 4. defining properties for associations

The resulting GFM based model is detailed in Clause 6.

5.10. WOS Approaches

The Web Object Server concept allows multi-faceted information to be stored, updated and queried so as to effectively allow the management and exploitation of the information in an intelligence way.

Two possible WOS approaches have been identified for evaluation in TB12:

- Based on CSW-ebRIM
- Based on WFS

Both the CSW and WFS support geographic, numeric, and textual queries based on the OGC Filter query language. Both also deliver data in response to query requests. In the case of the WFS, that data is in the form of GF_Features. The CSW typically returns "metadata" records. This distinction is one of use rather than substance. Starting with NAS version 7.0, NGA has re-integrated the NSG Metadata Foundation (NMF) back into the NAS. Metadata and Feature data become two views into

the same data holdings. In the same way, the WFS and CSW based WOSs can be seen as two different approaches to the same capability.

5.10.1. CSW ebRIM based WOS Approach

Information Types Supported

The CSW ebRIM based approach recognizes that it is unlikely that there is a single ‘store’ type capable of storing multi-faceted data. For the purposes of this document multi-faceted can be considered at least:

- Structured Geospatial Data, Vector and Gridded (defined by a schema)
- Unstructured Geospatial Data, add-hoc Attributes and Object Types.
- Tabular Data
- Video and Imagery (geo-referenced and non-geo-referenced)
- Relationship Data (associations, ad-hoc and characterised)
- Stream Objects (no-SQL)

Store Types

To store these artifacts at least the following database/web service are typically used:

- Relational Database/WFS
- Gridded Files/WCS
- Video File Store/H264 Stream + MISB KLV metadata streams
- Imagery File Store/WCS Service
- Sensor Information/SOS/STA
- Event Sources/ATOM+RSS+XMPP feeds
- RDF Store/SPARQL or GeoSPARQL Endpoint

Representation

In order to support the wide range of information required the following general principles will be adopted in designing the model.

Object Hierarchy

The Object store will support high level conceptual objects which may have multiple representations in detail, for example different geometries at different scales. Thus there is a premise that the ‘classic’ geo-feature represented by a geometry (e.g. a road polyline) with for example ‘width’ properties is an ‘feature style attribute’ of an object. We could call these ‘SimpleFeature Attributes’ as an object could still be considered a Feature but the OGC SimpleFeature profile feature is more primitive.

Feature/Attribute Schema Independence

The proposed model to store the necessary information is an object hierarchy based on Ontological principles rather than schema principles. The main conceptual difference is to break the specific link between feature and attribute, avoiding constraining an object to have a fixed set of attributes. An object can be 'classed' by associating one or more classes with it, but these do not define the attribute profile. Note this also allows an instance attribute to be used by more than one object.

Globally Unique Identifier Principle

In line with ontological principles a WOS object will have a well define characterizing global id (it can have more than one, but no one id can reference more than one object). In addition all WOS instance attributes will also have a global id. So in the example of road above, each simpleFeature object will have a globally unique id. This is critical in allowing an attribute to be retrieved from another store.

5.10.2. WFS based WOS Approach

Cubewerx is working on a WFS-based WOS, which would be a completely new service with an dedicated API. Presumably, this service would be able to handle a wide variety of object types including features, catalogue records, and even coverages. Like CSW or WFS it will use the Filter encoding standard as its predicate language. The underlying data model remains to be discussed, with ATOM being a favored candidate.

As a starting point the service proposal (02-049) will be used; it was derived from WFS/CSW and was the first attempt to move towards a unified data service (features, coverages, etc.).

5.11. Client Requirements

The goal of a WOS client is to enable the search, discovery and exploitation of the content provided by a WOS. In contrast to other geospatial web services, a WOS is intended to support a wider variety of types of objects. This section presents those requirements that were identified in the development of the WOS demonstrations.

The primary requirement for a WOS client is the ability to present both geospatial and non-geospatial content. An example of geospatial content in a geographically referenced Building. An example of non-geospatial content is a Decision. In Testbed-12 geospatial content was displayed both on a map and on a list, whereas non-geospatial content was presented on a list.

A first secondary requirement is an ability to present both text and non-text properties (i.e. multi media such as audio and video). An example of text properties is the name or length of a road, that is property values that can be encoded as text. Examples of non-text properties are a picture of a parcel or an audio clip.

Another secondary requirement is an ability to query and present associations between objects of content stored or registered in the WOS.

5.12. Initial Implementation Experiences

5.12.1. Data Availability

As of September 2016, all tweets of the 2015 Oakland March were still available via the Twitter search API. However we have not yet been able to find a definite answer to the lifetime of tweets. Whilst this might seem irrelevant to the monitoring of a live event, it has an impact on preserving the body of evidence for decisions. For the purpose of the GFM demo, it was therefore decided to harvest a selected number of tweets and store them in a backup system to emulate the live event. Deployed in a real-time scenario, this mechanism might also be relevant to the preservation of body of evidence data.

5.12.2. Data Georeferencing

An analysis of the tweets selected for the scenario revealed that only very few were actually georeferenced. However, georeferencing can be done based on keywords embedded in a tweet, e.g. “Front of the March now at <Fox Theatre>”, or identification of local features in pictures or video streams.

Chapter 6. The General Feature Model

The following paragraphs introducing the General Feature Model are based on [1], which even includes an introduction to object-oriented modeling and design. Readers not familiar with the object-oriented paradigm are referred advised to consult [1] first. ISO/TC211 follows the model driven architecture approach with its fundamental concept that an application can be described independently of the platform on which it runs. This level of abstraction results in the development of conceptual models, which are used to describe aspects of the real world in an abstract way. These models always reflect the specific view of the world by the model developer and contain only these elements from the universe of discourse that are relevant in a particular situation. Conceptual models have several levels of abstraction:

- **Meta-metamodel level**: The meta-metamodel contains the defining schema that identifies the principles to be used in organizing information about the concepts of interest.
- **Metamodel level**: The metamodel includes a specification of the language to be used in describing the concepts. A conceptual schema language provides the semantic and syntactic elements used in a rigorous description of the conceptual model consistent with the defining schema. ISO/TC211 elected UML the conceptual schema language
- **Application Level**: The application level contains application schemas that describe the specific concepts that are instantiated to produce a data set.
- **Lowest Level**: The lowest level is a data set that instantiates the concepts defined by an application schema.

6.1. Geographic Information: Key Principles

The defining schema (meta-metamodel) for geographic information contains several key principles:

- The real-world entities of interest for geographic information are described as **features**.
- Feature characteristics are known as **feature attributes**. These include spatial characteristics, which are of particular importance.
- Features may be related to each other in many ways with **relationships** upon spatial position being especially important.
- Features may perform a variety of functions in the natural or cultural environment in which they occur.
- A set of common characteristics, relationships, and functions common to many features define a **feature type**.

6.2. Metamodel for Geographic Information: The General Feature Model

The **General Feature Model** (GFM) is defined as the metamodel for the development of application schemas for geographic information. It is defined in ISO 19109. The principal element of the GFM is the metaclass *GF_FeatureType*. *Metaclass* is a stereotype that identifies a classifier whose instances are themselves classes, i.e. it represents a concept at a higher level of abstraction than do the

classes that instantiate it. An instance of GF_FeatureType is a UML class named for a specific feature type, such as road, alertMessage, or crowd.

As illustrated in the figure below, GF_FeatureType is connected to *GF_InheritanceRelation* by the associations *Specialization* and *Generalization*. Thus, two instances of *GF_FeatureType* may be related to each other as a superclass or a subclass in a generalization hierarchy.

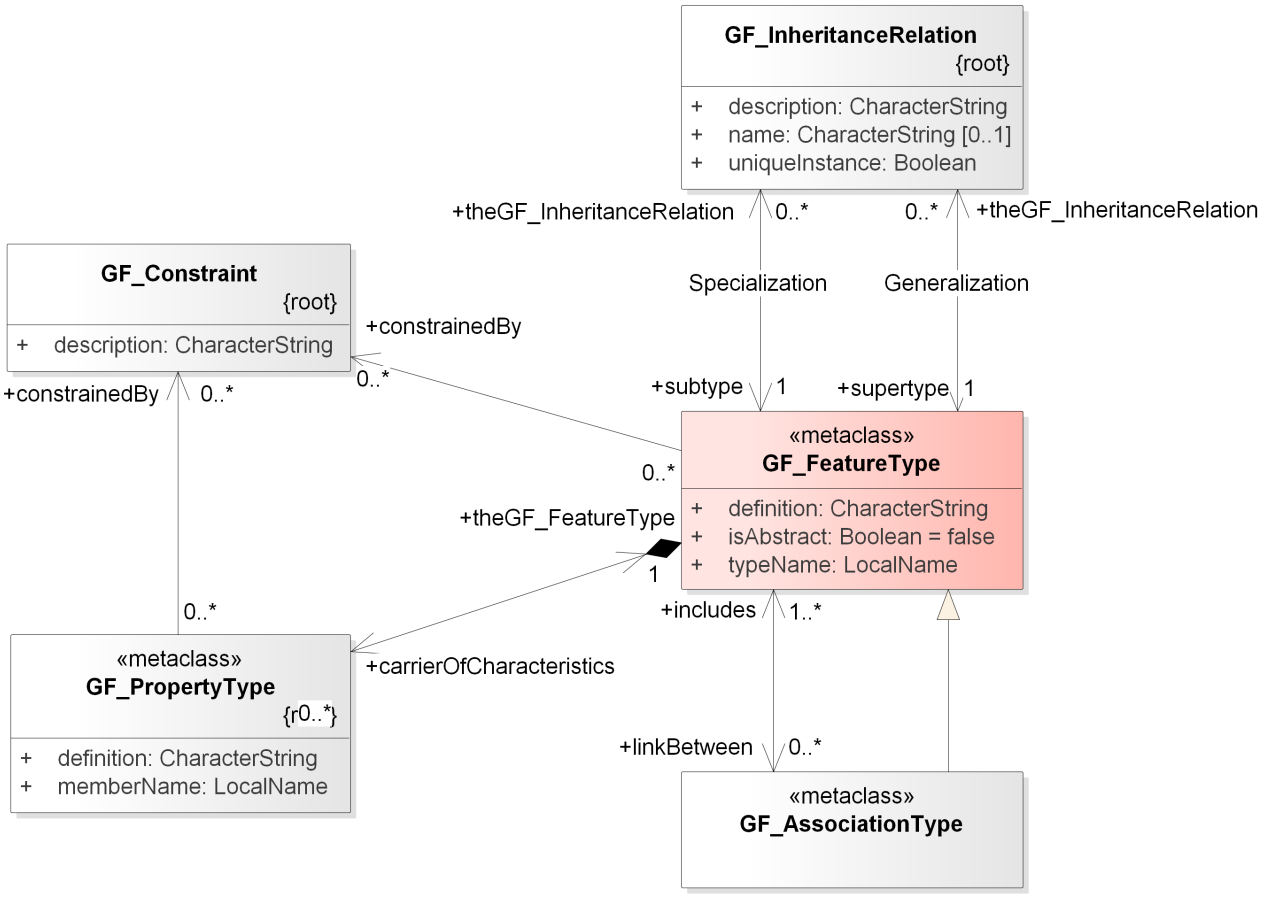

Figure 5. GF_FeatureType

The GFM defines two relationships between *GF_FeatureType* and *GF_AssociationType* as illustrated in figure above. The first association indicates that an instance of *GF_FeatureType* may be included in an instance of *GF_AssociationType*. Generally, this is implemented as a simple UML association between two classes in an application schema.

The second relationship between *GF_FeatureType* and *GF_AssociationType* is an inheritance relationship, i.e. *GF_AssociationType* inherits all associations and properties of *GF_FeatureType*, its superclass. This second relationship is modeled as an UML Association Class rather than a simple UML association in an application schema. Thus, a feature association can contain properties and - even more important - can be treated as a feature in its own right. The properties are shown by the composition between *GF_FeatureType* and *GF_PropertyType*. Both can be constrained, illustrated by the association to *GF_Constraint*. Constraints can be applied to any instance of its associated metaclasses.

GF_PropertyType is an abstract metaclass with three subclasses: GF_Operation, GF_AttributeType, and GF_AssociationRole as illustrated in figure below to define details about a feature's properties.

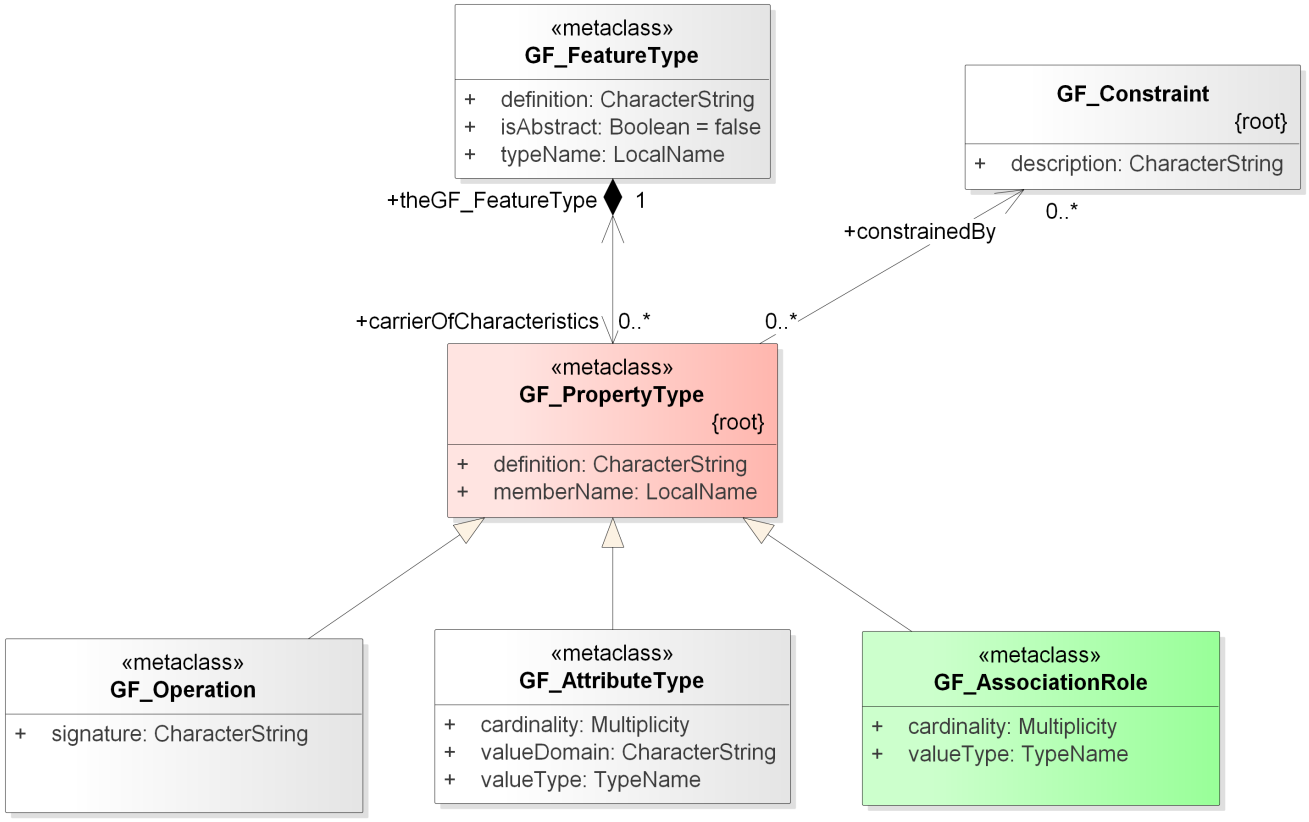

Figure 6. GF_PropertyType with subclasses. Highlighted: GF_AssociationRole to describe the role of an association

GF_AssociationRole describes a role that an instance of a feature type may have in an association with another instance of the same of another feature type. As illustrated in figure below, *GF_AssociationRole*'s *cardinality* attribute specifies the multiplicity for instances of the feature type acting in this role. It further inherits *GF_Constraint* from the GF_PropertyType to define further constraints.

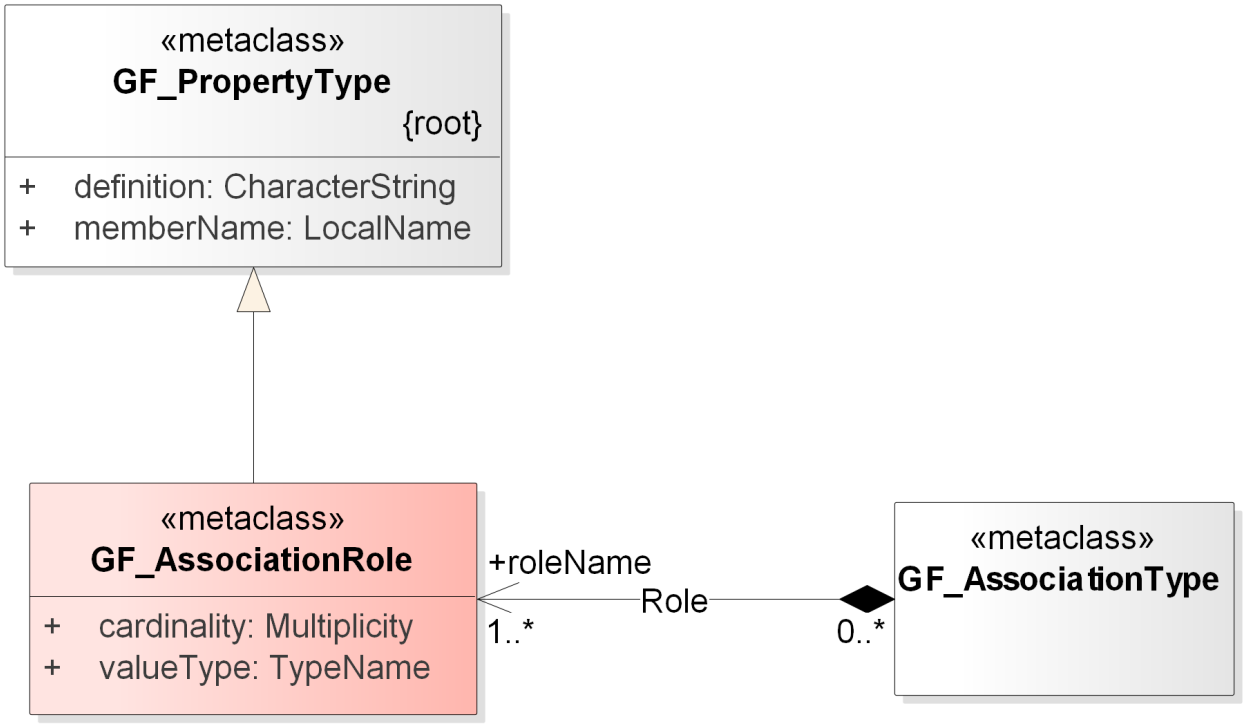

Figure 7. GF_AssociationRole

In summary, the fact that GF_AssociationType is a subclass of GF_FeatureType allows modeling any feature as a first class object, independently if the feature exists as a representation of a real world object, or is defined as part of an association between two features. Feature association types can be stored in registries or communicated independently of their associated feature types if necessary, which allows "show me all association types of this feature" type of requests.

In the example provided below, *BodyOfKnowledge* is modeled as a *GF_AssociationType* class that further defines the association between *Decision* and *CommunicationMessage*. As an instance of a GF_FeatureType specialization, it carries its own property (*trustLevel*).

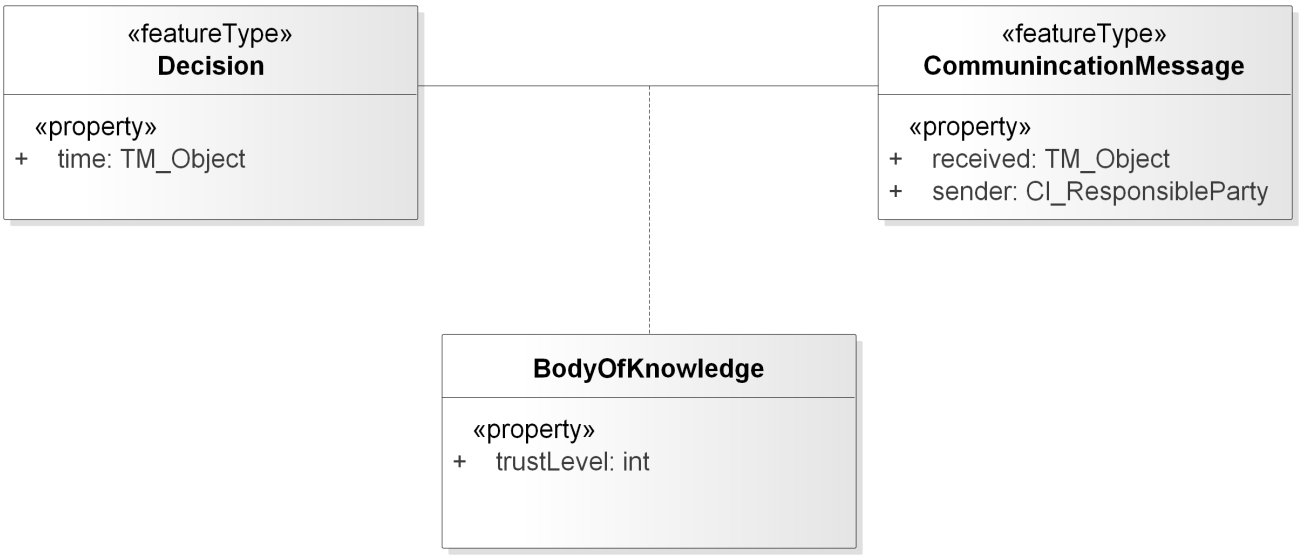

Figure 8. Example of an association as a first class object

Chapter 7. CSW-eb-RIM based WOS Implementation

There are a number of possible implementations of WOS using a CSW. The following section describes a two-tier approach used in Testbed-12. The ‘high level’ objects are stored in an overall service (the CSW-ebRIM registry). These high level objects have attributes, some simply values but others complex attributes such as very detailed geometries and properties against these. These attributes are ‘instanced’ attributes and are stored in other stores and available via web services, e.g. simpleFeatures, delivered by a WFS. It should be possible to retrieve any instance attribute using a well defined id query, for example a GetFeatureByID. A screenshot of the CSW-ebRIM registry used to implement the WOS Catalogue is shown below.

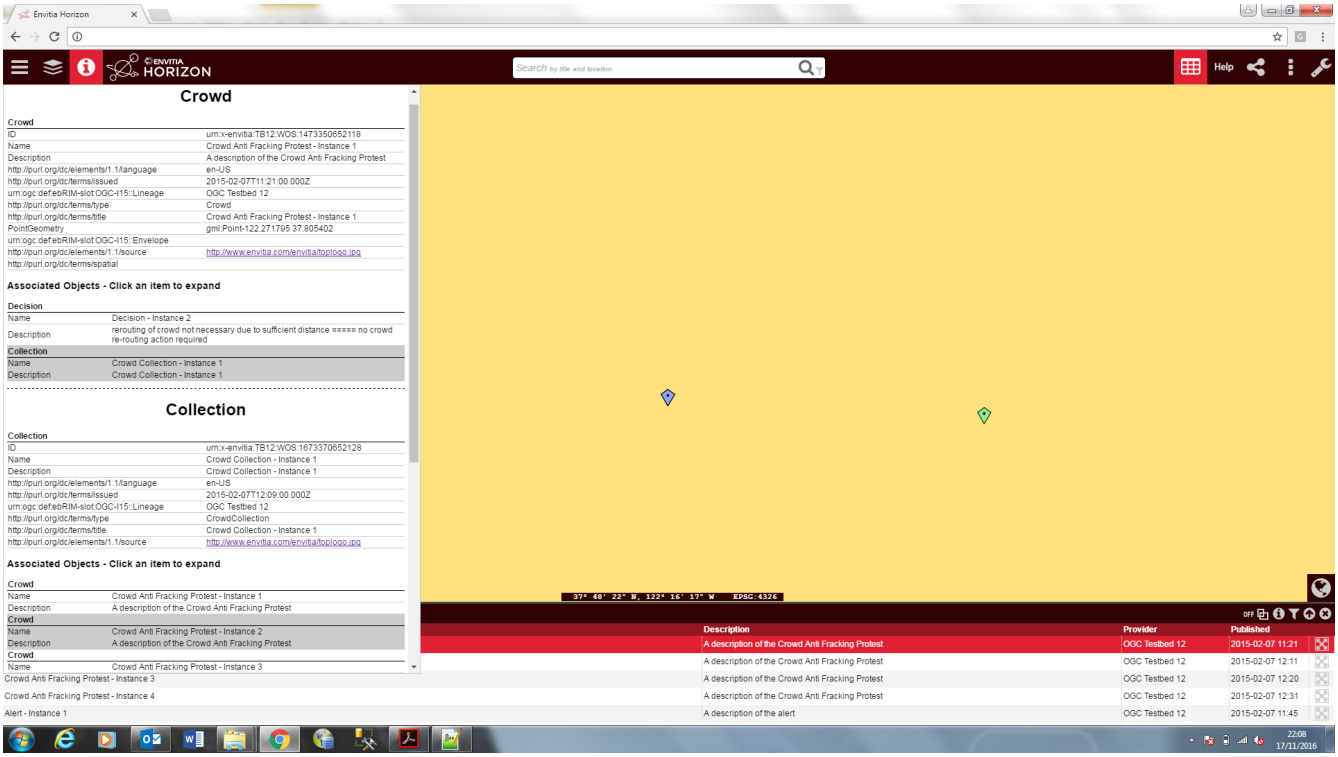

Figure 9. Object Composition

It is possible to abstract simple attributes, for example storing a geospatial extent of the geometric attribute in the registry. This allows queries on the high level objects such as ‘find me all the objects within a given area. A first level query can find the objects but a further detained query on each one is necessary to identify the exact set of objects meeting the criteria.

In this model the overall object server, really an object catalogue, provides what amounts to ‘metadata’ about the object, and the details of the object are dispersed among the various databases and services which support the ‘instance’ attributes of the object.

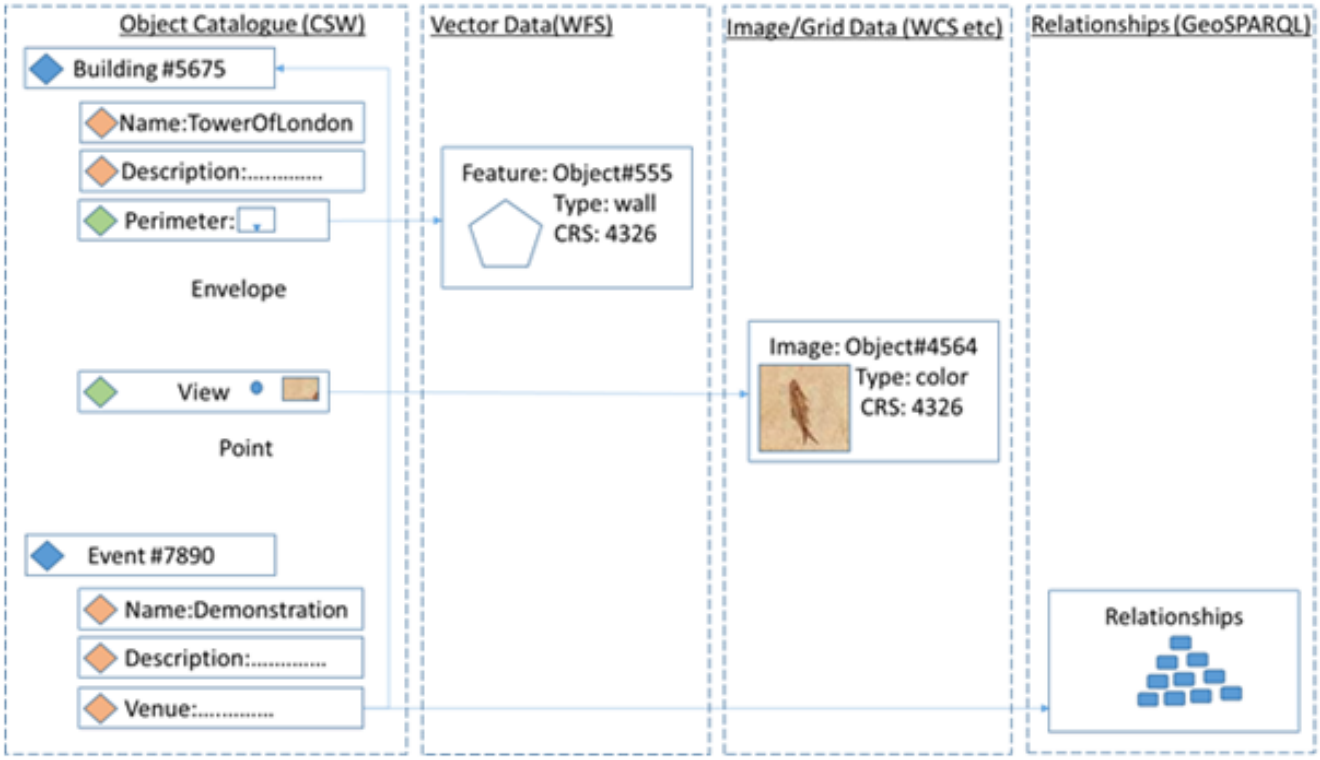

Figure 10. Object Composition

7.1. Interaction Model

The interaction (search and access model) with the WOS requires a relatively intelligent client that is able to query objects and the associations between them.

7.2. Searching for WOS Objects

To fully access an object it is necessary to first discover a high level WOS object, either by directly knowing its identifier or performing a search. When searching it is possible the search may be simple, for example find object by name. It could be a more complex query such as to find all objects of type ‘building’ in ‘area’. The latter can partially be resolved based on bounding box or point but not completely. Glossing over this for the moment, let us say we are happy with all ‘potential’ objects in the area then all is well. Lastly client application can ask for all buildings related to incident ‘7890’ and can find them all, although again this is a simplification. Each of the above queries can be satisfied using the ‘CSW-ebRIM’ protocol. We can see thumbnails of the objects via this interface too (given the above model).

7.3. Interrogation of a WOS Object

Having found an object the user may wish to dive into the detail, looking at the specific details of the perimeter, a particular image, or the detailed relationships, then it is necessary to use the appropriate query to one of the attribute servers (WFS, WCS, SPARQL endpoint etc).

The WOS Catalogue can help here, as each attribute instance described and associated to an object can have a class, and describes the query to be used to retrieve the object. i.e. the ‘id’ can be a fully populated query, for example a WFS Get request for a feature of id 555.

This model therefore does not necessarily require a client to understand the query model, simply to be able to execute the query specified. However the return result is of whatever type the service returns. In practice the client needs to understand the attribute delivery service for that attribute and decide if it can read that attribute type.

There is an argument that this is more complicated than a single WFS interface, but it is easily scalable. The WOS Catalogue allows addition of multiple database types, and there is no single query interface applicable to all datatypes so this model does offer a highly scalable approach. Testbed-12 demonstrated this capability by allowing a photograph and an audio clip to be served from an Accumulo database as the value of an attribute of an object registered in the WOS.

7.4. Object Insertion

This is to some degree complicated. Firstly attributes can exist in their own right, and so can be created independently (e.g. Transactional WFS, SPARQL etc). Also WOS objects are easy to create using an ebRIM Insert transaction. Attributes can also be easily related to a WOS using ebRIM if their ID is known (which it will be if they are created by the client, or searched for independently). When creating an attribute reference in an object it is essential that summary metadata is created, for example the extent of the perimeter object (an abstraction of the full geometry) needs to be populated to allow spatial searches.

7.5. Update and Deletion

When an attribute instance is updated, several actions are needed. If the attribute is accessed via a WOS Object, the WOS Object is known and any change to its summary metadata can be enacted. However we have said an attribute can be related to more than one WOS object. One approach is to search using an ebRIM query for all attributes with a global id (of the attribute instance) and update them all. The alternative is back-pointers but this is likely to get even more unwieldy. Similarly for deletion, if an object is deleted, all references to it need to be deleted. Again these processes put a load on the client, but some automation is ultimately possible.

Chapter 8. WFS based Web Object Service

8.1. Introduction

The Web Object Service (WOS) is a general data access service for objects. It can be considered an extension of a Web Feature Service where the restriction to operate on features has been relaxed.

The general characteristics of a WOS are:

- Provides a service description document (i.e. capabilities) describing:
  - the type of service being offered
  - information about the service provider
  - a set of parameter and service constraints
  - the list of object collections offered by the service
- access to general objects (i.e. members of a collection)
  - objects can be features
  - objects can also be binary objects
  - objects can be abstract (e.g. decisions)
- simple query capability
  - the ability to dynamically define collections of object using simple predicates (e.g. time, space, taxonomy)
- simple transaction capability
  - the ability to create/modify/delete single objects

This clause describes, in general terms, the CubeWerx implementation of a WOS for OGC Testbed 12.

8.2. Requirements

The following set of general requirements were used to drive the design and implementation of the WOS provided by CubeWerx:

1. A WOS can handle any kind of object including binary objects.
2. Objects are exposed through the WOS API using a common query model that includes spatial and/or temporal properties.
3. A WOS can maintain associations between objects under its control and other non-WOS resources
4. Associations can be simple links or they can be considered first class objects with properties.
5. A WOS can classify objects using any number of classification schemes.

6. A WOS query can include spatial, non-spatial, temporal and taxonomic predicates

8.3. Architecture

The following diagram illustrates the architecture on the CubeWerx WOS implemented for Testbed 12:

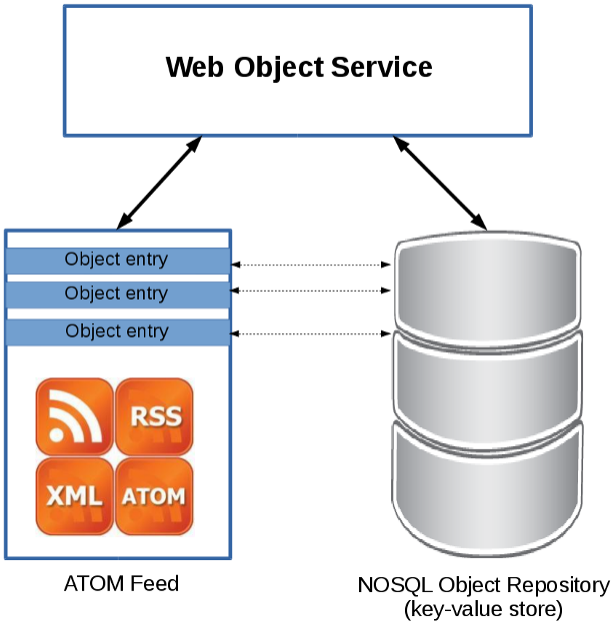

Figure 11. Architecture diagram of the CubeWerx's TB12 WOS

On the back end, the WOS service was a hybrid system composed of a RDBMS (i.e. Oracle 12c) used to manage the information model (i.e. associations, taxonomies and object metadata) of the WOS and a NoSQL key-value store (Cassandra 2.2) as the native object repository.

8.4. Basic Service Elements

8.4.1. Service Root

The service metadata document of a WOS is accessible at some service root URL. Additional resource URLs (i.e. hypermedia controls) can then be determined from the capabilities document.

The response to accessing the service root URL of a WOS with the HTTP GET methods shall be an XML-encoded, OGC capabilities document. The service root URL is opaque and its form is not defined in this engineering report.

8.4.2. Object Representation

Objects managed by a WOS have a native representation (i.e. text, video, WORD document, etc.) and a representation used by the WOS for query purposes (i.e. the query model). The canonical representation of the WOS' query model is an XML-encoded, ATOM feed.

Other query models may be supported, include deep-searches of the content itself but for Testbed 12, the only supported query model was the ATOM feed.

8.4.3. Content Negotiation

Two common approaches for content negotiation are supported by the WOS; HTTP headers (i.e. Accept header) and HTTP query parameters (i.e. outputFormat or f). In any context there must be agreement between content negotiation methods used; otherwise an exception shall be raised (HTTP code=415). Content negotiation using the Accept header shall proceed in the standard manner describe in the HTTP protocol:

- When submitting an HTTP request to a service, the client shall set the Accept header with a list of desired response content types in order of preference
- The server shall respond with the most preferred content type it supports
- An exception shall be thrown if none of the specified content types are supported
- In any context, the HTTP OPTIONS method may be used to get the list of support content types. (NOTE: this is not implemented for Testbed 12)

Content negotiation using the HTTP query parameters shall proceed in the usual OGC way (i.e. the desired content type is specified using the output format query parameter and the server either supports it or it does not). The list of supported formats for the query model of a WOS shall be advertised in the server's capabilities document.

8.4.4. Exceptions

When a service exception occurs, the server shall respond with an appropriate HTTP error code. All exceptions that are the result of an invalid client request should have an HTTP error code in the 4xx range (and in most, if not all cases, should specifically be "400 Bad Request"). Server-side issues should be given an HTTP error code in the 5xx range (and in most if not all cases) should specifically be "500 Internal Server Error".

The canonical content of an exception shall be an OGC service exception report. This engineering report does not define behavior for all HTTP methods for all resource (e.g. the behavior of the PUT method is not described for the capabilities resource "/"). When a server is presented with an unsupported HTTP method for a particular resource it shall respond with the HTTP error code "405 Method Not Allowed" and should contain an OGC exception message in the body of the response.

NOTE Implementations are free to implement (or not) whatever behavior they want in cases where behavior is not defined in this engineering report (e.g. for experimental purposes or vendor extensions) but whatever they do will be outside the scope of this engineering report.

8.5. Authentication and Access Control

All methods and resource described in the engineering report are subject to authentication and access control. It is envisioned that authentication and access control will be layered on top of a RESTful WOS but the description of how that would be done is beyond the scope of this engineering report.

NOTE For OGC Testbed 12, the CubeWerx WOS can be protected using HTTP Basic Authentication.

8.6. Hypermedia Controls

8.6.1. Introduction

Hypermedia controls are the means by which, at any given state, a server enumerate for client what possible next states are available.

Hypermedia controls defined by a WOS shall be encoded, in XML, as ATOM links. ATOM links are defined in the following schema:

- http://schemas.opengis.net/kml/2.2.0/atom-author-link.xsd

The canonical value for the rel attribute shall either be:

- a token taken from the IANA Link Relation register (see: http://www.iana.org/assignments/link-relations/link-relations.xhtml)
- or it shall be a URI registered with the OGC Naming Authority
- or it shall be a value defined in this engineering report that shall be submitted to the OGC NA for registration (eventually)

8.6.2. Hypermedia Control in a WOS' Capabilities Document

The following hypermedia controls shall be included in the capabilities document of a WOS:

- for each offered object collection listed in the capabilities document:
  - a link, with rel="collection", shall be included that points to the root URL of the collection
- a server is free to include any other desired hypermedia controls in the server's capabilities document

8.6.3. Hypermedia Controls in a WOS' Query Response

The following hypermedia control shall be included in the response to a WOS query:

- the query response element (i.e. ATOM entry) shall contain a link with rel="self" containing the entry's URL

A server is free to include any other hypermedia controls within an ATOM entry. Some examples of common links that may be used include:

- alternate: pointing to alternate representations of the ATOM entity
- edit: pointing to a location where the entry can be updated or deleted
- next/prev: pointing to a related next or previous object within a chain of resources

8.7. Summary of Resources

The following tables summarizes the resources offered by a WOS:

Table 2. Summary of WOS resources

Resource Class	Description	Access path
Capabilities	The complete service metadata document	/
Object collection	A collection of objects	/{collection name}
Object	An object (i.e. a specific member of a named collection)	/{object URL}
Association	A directional relationship between two objects	/associations[/{id}]
ClassificationScheme	A taxonomy	/classificationSchemes[/{id}]
ClassificationNode	A node in a taxonomy	/classificationNodes[/{id}]
Classification	An association between an object and a node in a taxonomy	/classifications[/{id}]

8.8. MIME Types

The following MIME types are defined by this engineering report:

- Associations document
  - encoded as XML: application/ogc-as+xml
- Classification document:
  - encoded as XML: application/ogc-cl+xml

8.9. A Word about Examples

Many of the descriptions in this engineering report are accompanied by examples in the form of sequence diagrams showing the HTTP interaction between a client and a server. These examples reference a fictitious service root: http://www.someserver.com/1.0/wos/. The "1.0/wos" component of the path is the examples is not part of the canonical resource URLs but simply part of this example's service root.

In a number of examples, normally packed text that has been wrapped across multiple lines for the sake of clarity and space.

In many cases, the XML does not contain all the necessary information in order to validate; again, this is done for the sake of clarity.

8.10. Service Metadata

8.10.1. Introduction

A WOS' capabilities document includes hypermedia controls to get the URL of each object collection offered by the WOS and a URL endpoint that allows objects to be added, modified or removed from the collection.

8.10.2. Resources

The base or root URL of a server (path = '/') shall resolve to the service metadata document (a.k.a. the OGC Capabilities document).

8.10.3. Query Parameters

The following table lists the query parameters that may be used with the capabilities resource:

Table 3. Capabilities query parameters

Parameter Name	Description
acceptVersions	A comma-separated list of acceptable version ordered most preferable first.
sections	A comma-separated list of section names to be included in a capabilities document. Valid values are: ServiceIdentification, ServiceProvider, OperationsMetadata, ObjectCollections, FilterCapabilities.
acceptFormats	A comma-separated list of acceptable formats for the capabilities document order most preferable first.

8.10.4. Representations

The canonical representation for a service metadata document shall be the XML-encoded OGC capabilities document as defined in the wos.xsd schema.

NOTE	The wos.xsd schema may be accessed at online at http://cubewerx.pvretano.com/schemas/wos/1.0.0.

The recommended other output format for a capabilities document is HTML5. The content of the HTML5 version of the capabilities document shall be the same a the XML version. This engineering report does not provide a standard definition for a capabilities document expressed in HTML5. Implementations are free to make the HTML5 output look as pretty as they like.

Other representations (e.g. JSON) are allowed but are not described in this engineering report.

8.10.5. Methods

The following table describes the actions that server should take when the specified method is applied to the specified resource. The resources from

Table 4. Methods

Resource	Method	Action
/	GET	Gets the complete capabilities document
/	POST	Not specified by this engineering report.
/	PUT	Not specified by this engineering report.
/	DELETE	Not specified by this engineering report.

8.10.6. Examples

Example 1: Get the list of object collections offered by a WOS

```
   CLIENT                                                    SERVER
     |                                                          |
     |   GET /1.0/wos?sections=ObjectCollections  HTTP/1.1      |
     |   Host: www.someserver.com                               |
     |   Accept: text/xml                                       |
     |--------------------------------------------------------->|
     |                                                          |
     |   HTTP/1.1 200 OK                                        |
     |   Content-Type: text/xml                                 |
     |                                                          |
     |   <?xml version="1.0"?>                                  |
     |   <ObjectCollections...>                                 |
     |      <ObjectCollection>                                  |
     |         <atom:link rel="collection"                      |
     |                    type="application/atom+xml"           |
     |                    href="http://www.someserver.com/wos/1.0/objects"/>
     |         <atom:link rel="edit"                            |
     |                    href="http://www.someserver.com/wos/1.0/objects"/>
     |         <Name>Objects</Name>                             |
     |         <Title>The Default Object Collection</Title>     |
     |         <Abstract>This is the default collection used by the WOS to manage
objects</Abstract>
     |         <ows:Keywords>                                   |
     |            <ows:Keyword>WOS</ows:Keyword>                |
     |            <ows:Keyword>web</ows:Keyword>                |
     |            <ows:Keyword>object</ows:Keyword>             |
     |            <ows:Keyword>service</ows:Keyword>            |
     |         </ows:Keywords>                                  |
     |         <DefaultCRS>urn:ogc:def:crs:EPSG::6269</DefaultCRS>
     |         <OtherCRS>urn:ogc:def:crs:EPSG::32615</OtherCRS>  |
     |         <OtherCRS>urn:ogc:def:crs:EPSG::32616</OtherCRS>  |
     |         <OtherCRS>urn:ogc:def:crs:EPSG::32617</OtherCRS>  |
     |         <OtherCRS>urn:ogc:def:crs:EPSG::32618</OtherCRS>  |
     |         <ows:WGS84BoundingBox>                           |
     |            <ows:LowerCorner>-180 -90</ows:LowerCorner>   |
     |            <ows:UpperCorner>180 90</ows:UpperCorner>     |
     |         </ows:WGS84BoundingBox>                          |
     |      </ObjectCollection>                                 |
     |   </ObjectCollections>                                   |
     |<---------------------------------------------------------|
```

Example 2: Get the complete capabilities document

```
   CLIENT                                                    SERVER
     |                                                          |
     |    GET /wos/1.0   HTTP/1.1                               |
     |    Host: www.someserver.com                              |
     |    Accept: text/xml                                      |
     |--------------------------------------------------------->|
```

```
|                                                                  |
|    HTTP/1.1 200 OK                                               |
|    Content-Type: text/xml                                        |
|                                                                  |
|    <?xml version="1.0" encoding="UTF-8"?>                        |
|    <WOS_Capabilities                                             |
|       version="1.0.0"                                            |
|       xmlns="http://www.opengis.net/wos/1.0"                     |
|       xmlns:gml="http://www.opengis.net/gml/3.2"                 |
|       xmlns:atom="http://www.w3.org/2005/Atom"                   |
|       xmlns:fes="http://www.opengis.net/fes/2.5"                 |
|       xmlns:xlink="http://www.w3.org/1999/xlink"                 |
|       xmlns:ows="http://www.opengis.net/ows/2.0"                 |
|       xmlns:xsd="http://www.w3.org/2001/XMLSchema"               |
|       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
|       xsi:schemaLocation="http://www.opengis.net/wos/1.0         |
|           http://schemas.opengis.net/wos/1.0/wos.xsd             |
|           http://www.opengis.net/ows/2.0                         |
|           http://schemas.opengis.net/ows/2.0/owsAll.xsd          |
|           http://www.w3.org/2005/Atom                            |
|           http://schemas.opengis.net/kml/2.2.0/atom-author-link.xsd">
|       <ows:ServiceIdentification>                                |
|          <ows:Title>OGC Member WOS</ows:Title>                   |
|          <ows:Abstract>                                          |
|             Web Feature Service maintained by NSDI data          |
|             provider, serving FGDC framework layer XXX;          |
|             contact Paul.Bunyon@BlueOx.org                       |
|          </ows:Abstract>                                         |
|          <ows:Keywords>                                          |
|           <ows:Keyword>WOS</ows:Keyword>                         |
|           <ows:Keyword>web</ows:Keyword>                         |
|           <ows:Keyword>object</ows:Keyword>                      |
|           <ows:Keyword>service</ows:Keyword>                     |
|           <ows:Type>String</ows:Type>                            |
|          </ows:Keywords>                                         |
|          <ows:ServiceType>WOS</ows:ServiceType>                  |
|     <ows:ServiceTypeVersion>1.0.0</ows:ServiceTypeVersion>|
|     <ows:Fees>NONE</ows:Fees>                                    |
|     <ows:AccessConstraints>NONE</ows:AccessConstraints>          |
|       </ows:ServiceIdentification>                               |
|       <ows:ServiceProvider>                                      |
|          <ows:ProviderName>CubeWerx Inc.</ows:ProviderName> |
|          <ows:ProviderSite                                       |
|               xlink:href="http://www.cubewerx.com"/>            |
|          <ows:ServiceContact>                                    |
|             <ows:IndividualName>                                 |
|                Panagiotis (Peter) A. Vretanos                    |
|             </ows:IndividualName>                                |
|             <ows:PositionName>                                   |
|                Senior developer                                  |
|             </ows:PositionName>                                  |
```

```
|             <ows:ContactInfo>                                  |
|                <ows:Phone>                                     |
|              <ows:Voice>1.800.555.1212</ows:Voice>             |
|              <ows:Facsimile>1.800.555.1313</ows:Facsimile>     |
|                </ows:Phone>                                    |
|                <ows:Address>                                   |
|                   <ows:DeliveryPoint>                          |
|                      123 Main St.                              |
|                   </ows:DeliveryPoint>                         |
|                   <ows:City>Toronto</ows:City>                 |
|                   <ows:PostalCode>12345</ows:PostalCode>       |
|                   <ows:Country>Canada</ows:Country>            |
|                   <ows:ElectronicMailAddress>                  |
|                      pvretano@cubewerx.com                     |
|                   </ows:ElectronicMailAddress>                 |
|                </ows:Address>                                  |
|             </ows:ContactInfo>                                 |
|             <ows:Role>PointOfContact</ows:Role>                |
|          </ows:ServiceContact>                                 |
|       </ows:ServiceProvider>                                   |
|       <ows:OperationsMetadata>                                 |
|          <ows:Operation name="GetCapabilities">                |
|             <ows:DCP>                                          |
|                <ows:HTTP>                                      |
|                   <ows:Get                                     |
|        xlink:href="http://www.someserver.com/wos/1.0?"/>       |
|                </ows:HTTP>                                     |
|             </ows:DCP>                                         |
|          </ows:Operation>                                      |
|          <ows:Operation name="NoOp">                           |
|             <ows:DCP>                                          |
|                <ows:HTTP>                                      |
|                   <ows:Get                                     |
|        xlink:href="http://www.someserver.com/wos/1.0?"/>       |
|                </ows:HTTP>                                     |
|             </ows:DCP>                                         |
|          </ows:Operation>                                      |
|          <!-- ************************************** -->|
|          <!-- *   CONFORMANCE DECLARATION           * -->|
|          <!-- ************************************** -->|
|          <ows:Constraint name="WOS">                           |
|             <ows:NoValues/>                                    |
|             <ows:DefaultValue>TRUE</ows:DefaultValue>          |
|          </ows:Constraint>                                     |
|          <!-- ************************************** -->|
|          <!-- *   CAPACITY CONSTRAINTS              * -->|
|          <!-- ************************************** -->|
|          <ows:Constraint name="CountDefault">                  |
|             <ows:NoValues/>                                    |
|             <ows:DefaultValue>1000</ows:DefaultValue>          |
|          </ows:Constraint>                                     |
```

```
|          <ows:Constraint name="ResponseCacheTimeout">     |
|             <ows:NoValues/>                               |
|             <ows:DefaultValue>300</ows:DefaultValue>      |
|          </ows:Constraint>                                |
|          <ows:Constraint name="AutomaticDataLocking">     |
|             <ows:NoValues/>                               |
|             <ows:DefaultValue>TRUE</ows:DefaultValue>     |
|          </ows:Constraint>                                |
|       </ows:OperationsMetadata>                           |
|       <ObjectCollections>                                 |
|          <ObjectCollection>                               |
|             <atom:link rel="collection"                   |
|                        type="application/atom+xml"        |
|                        href="http://.../objects"/>        |
|             <atom:link rel="edit"                         |
|                        href="http://.../objects"/>        |
|             <Name>Objects</Name>                          |
|             <Title>The Default Object Collection</Title>  |
|             <Abstract>                                    |
|                The default collection used by this WOS    |
|                to manage objects.                         |
|             </Abstract>                                   |
|             <ows:Keywords>                                |
|                <ows:Keyword>WOS</ows:Keyword>             |
|                <ows:Keyword>web</ows:Keyword>             |
|                <ows:Keyword>object</ows:Keyword>          |
|                <ows:Keyword>service</ows:Keyword>         |
|             </ows:Keywords>                               |
|             <DefaultCRS>urn:ogc:def:crs:EPSG::6269</DefaultCRS>
|             <OtherCRS>urn:ogc:def:crs:EPSG::32615</OtherCRS>
|             <OtherCRS>urn:ogc:def:crs:EPSG::32616</OtherCRS>
|             <OtherCRS>urn:ogc:def:crs:EPSG::32617</OtherCRS>
|             <OtherCRS>urn:ogc:def:crs:EPSG::32618</OtherCRS>
|             <ows:WGS84BoundingBox>                        |
|                <ows:LowerCorner>-180 -90</ows:LowerCorner>|
|                <ows:UpperCorner>180 90</ows:UpperCorner>  |
|             </ows:WGS84BoundingBox>                       |
|          </ObjectCollection>                              |
|       <fes:Filter_Capabilities>                           |
|           ...                                             |
|             <fes:SpatialOperators>                        |
|                <fes:SpatialOperator name="BBOX"/>         |
|                <fes:SpatialOperator name="Equals"/>       |
|                <fes:SpatialOperator name="Disjoint"/>     |
|                <fes:SpatialOperator name="Intersects"/>   |
|                <fes:SpatialOperator name="Touches"/>      |
|                <fes:SpatialOperator name="Crosses"/>      |
|                <fes:SpatialOperator name="Within"/>       |
|                <fes:SpatialOperator name="Contains"/>     |
|                <fes:SpatialOperator name="Overlaps"/>     |
|                <fes:SpatialOperator name="Beyond"/>       |
```

```
|               <fes:SpatialOperator name="DWithin"/>      |
|            </fes:SpatialOperators>                       |
|          ...                                             |
|            </fes:TemporalOperands>                       |
|            <fes:TemporalOperators>                       |
|            <fes:TemporalOperator name="After"/>          |
|            <fes:TemporalOperator name="Before"/>         |
|            <fes:TemporalOperator name="Begins"/>         |
|            <fes:TemporalOperator name="BegunBy"/>        |
|            <fes:TemporalOperator name="TContains"/>      |
|            <fes:TemporalOperator name="During"/>         |
|            <fes:TemporalOperator name="TEquals"/>        |
|            <fes:TemporalOperator name="TOverlaps"/>      |
|            <fes:TemporalOperator name="Meets"/>          |
|            <fes:TemporalOperator name="OverlappedBy"/>   |
|            <fes:TemporalOperator name="MetBy"/>          |
|            <fes:TemporalOperator name="EndedBy"/>        |
|            </fes:TemporalOperators>                      |
|          ...                                             |
|    </WOS_Capabilities>                                   |
|<---------------------------------------------------------|
```

8.11. Object Access & Management (GetObject/Transaction)

8.11.1. Introduction

This clause describes how objects from a collection can be accessed, added, modified or deleted.

8.11.2. Resources

The following table describes the object resources offered by a WOS.

Table 5. Summary of WOS resources

Resource Class	**Description**	**Access path**
Object collection	A collection of objects	/{collection name}
Object	An object (i.e. a specific member of a named collection). The URL for an object is returned the an object is added to a collection. The URL for an object may also be determined from the response to a query (i.e. the rel="self" link)	/{object URL}

All web object server implementations shall provide at least one object collection named *objects*.

8.11.3. Query parameters

Query parameters may be appended to an object URL to control the output format and content of a query response and identify a subset of objects from a collection. The following table defines the query parameters that may be appended to an object URL.

Table 6. Object query parameters

Parameter	Description
outputFormat	Specifies the representation of the query response
count	Specifies the number of objects to present in the query response
startIndex	Specifies the starting point from which objects are presented in the query response
namespaces	Used to declare the namespaces for qualified names
srsName	The CRS to be used when presenting geometric values in the query response
filterLanguage	Specifies the language used to define a filter
filter	Specifies a query filter in the language specified using the filterLanguage parameter
bbox	Specifies a bounding box predicate
geometry	A WKT-encoded geometry
crs	The coordinate reference system for the value of the geometry parameter
spatialOp	The spatial operation to use for testing the geometry of an object against the value of the geometry parameter. Valid values are: Equals, Disjoint, Touches, Withing, Overlaps, Crosses, Intersects (default), Contains
time	An ISO8601 time instance or interval
temporalOp	The temporal operator to apply when comparing an objects temporal property with the value of the time parameter. Valid values are: After, Before, Begins, During (default), EndedBy, Ends, TEquals, Meets, MetBy
classifiedAs	The identifier of a node within a classification scheme
classificationScope	One of narrow, exact or board

8.11.4. Representations

Each object managed by the WOS has two representations; a native representation and a corresponding ATOM representation used as the query model by the WOS.

The native representation is whatever representation is presented to the WOS when the object is

created (e.g. video format, image, MSWORD document, etc.). The native representation is stored and can be retrieved with full fidelity.

The canonical representation for the object's query model is the ATOM entry as described in RFC 5023. When querying a WOS, it is this representation that is queried to identify the set of objects that satisfy the query predicate.

8.11.5. Methods

The following table describes the actions the server should take when the specified method is applied to the specified resource.

Table 7. Methods

Resource	Method	Action
/{collection name}	GET	Get a subset of objects from the specified object collection. The number of objects returned depends on the value of the *count* query parameter.
	POST	Adds a new member to the object collection. A representation of the object is supplied as the body of the request.
	PUT	Replaces one or more objects in the specified collection. A representation of the new object is supplied in the body of the request.
	DELETE	Deletes one or more objects from the specified collection
/{object URL}	GET	Gets the specified object.
	POST	Not specified by this engineering report.
	PUT	Replaces the specified object. A representation of the replacement object is supplied om the body of the request.
	DELETE	Deletes all representations of the specified object

As previously *mentioned* each object managed by the WOS has its native representation and a corresponding representation that is used as the query model (i.e. an ATOM entry). A new object can be added to a collection in one of three ways.

- An ATOM entry is POSTed to the WOS that encodes the object in-line as a base64 string
- An ATOM entry is POSTed to the WOS and then that entry is subsequently updated, using the HTTP PUT method, to add the native representation of the object
- The native representation of the object is POSTed to WOS which stores that object and creates a minimal corresponding ATOM entry for the object to be used as the query model. This minimal ATOM entry may be subsequently updated, using the HTTP PUT method, to enrich its content.

8.11.6. Examples:

Example 1: Get a subset of objects from a collection

```
CLIENT                                                    SERVER
  |                                                         |
  |   GET /1.0/wos/object?count=27 HTTP/1.1                 |
  |   Host: www.someserver.com                              |
  |   Accept: application/atom+xml                          |
  |-------------------------------------------------------->|
  |                                                         |
  |   HTTP/1.1 200 OK                                       |
  |   Content-Type: application/atom+xml                    |
  |                                                         |
  |   <?xml version="1.0"?>                                 |
  |   <feed xmlns="http://www.w3.org/2005/Atom"             |
  |         xmlns:georss="http://www.georss.org/georss"     |
  |         xmlns:gml="http://www.opengis.net/gml"          |
  |         xmlns:cw="http://www.cubewerx.com/cw">          |
  |      <title>Web Object Server Default Collection</title>|
  |      <link rel="self"                                   |
  |            href="http://www.someserver.com/.../objects"/>|
  |      <updated>2016-07-30T03:18:16Z</updated>            |
  |      <subtitle>This is the default collection that the Web|
  |      Object Service uses for managing objects.  Specific|
  |      deployments of the server many have other object   |
  |      collections as well.</subtitle>                    |
  |      <id>urn:uuid:2e2c23df-a8db-4318-b52a-8b8efffcbb74</id>
  |      <generator uri="http://www.someserver.com/..."/>   |
  |      CubeWerx Web Object Service</generator>            |
  |      <entry cw:id="CWFID.OBJECTS.0.0">                  |
  |         <author>                                        |
  |            <name>Panagiotis Vretanos</name>             |
  |            <email>pvretano@cubewerx.com</email>         |
  |         </author>                                       |
  |         <content type="text/plain">Just some dummy content</content>
  |         <id>urn:uuid:ce28abbe-d17e-426d-8dae-a7977442a7be</id>
  |         <link href="http://www.someserver.com/..."      |
  |               type="video/mp4" rel="alternate"/>        |
  |         <link href="http://www.someserver.com/.../objects/CWFID.OBJECTS.0.0"
  |               type="application/atom+xml" rel="self"/>  |
  |         <summary>Example dummy object</summary>         |
  |         <title>Update an object</title>                 |
  |         <updated>20110118011435</updated>               |
  |         <georss:where>                                  |
  |            <gml:Point>                                  |
  |               <gml:pos>44.176426731297 -71.422232532932</gml:pos>
  |            </gml:Point>                                 |
  |         </georss:where>                                 |
  |      </entry>                                           |
  |   </feed>                                               |
  |<--------------------------------------------------------|
```

NOTE	The ATOM entry in this example includes a link with rel="alternate" that may be used to retrieve the native representation of the object.

Example 2: Get the subset of objects that lie within a specified bounding box.

```
CLIENT                                                    SERVER
  |                                                         |
  |   GET /1.0/wos/objects?bbox=10,10,20,20 HTTP/1.1        |
  |   Host: www.someserver.com                              |
  |   Accept: application/atom+xml                          |
  |-------------------------------------------------------->|
  |                                                         |
  |   HTTP/1.1 200 OK                                       |
  |   Content-Type: application/atom+xml                    |
  |                                                         |
  |   <?xml version="1.0"?>                                 |
  |   <feed> ... </feed>                                    |
  |<--------------------------------------------------------|
```

Example 3: Add a new object to a collection.

```
Step 1: Create the ATOM entry for the object.
```

```
CLIENT                                                    SERVER
  |                                                         |
  |   POST /1.0/wos/objects   HTTP/1.1                      |
  |   Host: www.someserver.com                              |
  |   Content-Type: application/atom+xml                    |
  |                                                         |
  |   <?xml version="1.0"?>                                 |
  |   <entry>                                               |
  |      .                                                  |
  |      .  An ATOM entry describing the object...          |
  |      .                                                  |
  |     <georss:where>                                      |
  |        <gml:Point>                                      |
  |           <gml:pos>44.176426731297 -71.422232532932</gml:pos>
  |        </gml:Point>                                     |
  |     </georss:where>                                     |
  |   </entry>                                              |
  |-------------------------------------------------------->|
  |                                                         |
  |   HTTP/1.1 201 Created                                  |
  |   Location: /1.0/wos/object/1014                        |
  |<--------------------------------------------------------|
```

```
Step 2: Update the object to add the native representation
```

Example 4: Delete an object

8.12. Associations

8.12.1. Introduction

Associations are use to relate objects managed in a WOS to each other and to other OGC and non-OGC resources. The basic structure of an association is:

<source identifier> <relationship> <target identifier>

The source and target identifiers reference the objects or resources that are participating in the association. The relationship or association type is an identifier that describes the relationship. For Testbed 12 relationships are either encoded as tokens taken from the IANA Link Relation register or are defined in the Web Integration Service engineering report (see OGC 16-043 & http://www.iana.org/assignments/link-relations/link-relations.xhtml).

8.12.2. Resources

The collection of associations is accessed via a resource with the path /associations.

A specific association is accessed via the URL provided by the system when the association was

created. This URL is normally opaque but in the case of CubeWerx's TB12 WOS, the URL template for an association is /associations/{id}.

Query Parameters

The following table summarizes the query parameters that may be used with the /associations resource. These intent of these parameters is to provide a simple query API for associations. The values specified for these parameters are logically combined using the *AND* operation to form a query predicate for selecting which associations should be presented in a response document.

Table 8. Object query parameters

Parameter	Description
outputFormat	Specifies the output representation of an association
id	The identifier for a specific association
rel	A relationship identifier (i.e. IANA token or OGC URN/URL)
sourceId	A source identifier or base URL of an OGC service
localSourceId	The local identifier for a source OGC resource
sourceType	The source object type
targetId	A target identifier or base URL of an OGC service
localTargetId	The local identifier for a target OGC resource
targetType	The target object type

Representations

The following XML schema fragment defines the XML-encoding for an Association and a collection of associations.

```
<xsd:element name="Associations">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element ref="wos:Association"
                      minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="version" type="xsd:string"
                     use="required" fixed="1.0.0"/>
   </xsd:complexType>
</xsd:element>

<xsd:element name="Association" type="wos:AssociationType"/>
<xsd:complexType name="AssociationType">
   <xsd:complexContent>
      <xsd:extension base="wos:BaseElementType">
         <xsd:sequence>
            <xsd:element name="Source" type="wos:ResourceType"/>
            <xsd:element name="Target" type="wos:ResourceType"/>
         <xsd:attribute name="id"
                        type="wos:TokenOrAnyURI" use="optional"/>
         <xsd:attribute name="rel"
                        type="wos:TokenOrAnyURI" use="required"/>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
```

This schema defines two classes of associations:

- Simple associations composed of source, target and relationship identifiers
- Associations with properties composed of source, target and relationship identifiers and additional, dynamically defined, properties

As shown in the schema above, associations can have identity that makes the management of associations more convenient and also allows associations between associations and other resources (including other associations) to be defined.

For Testbed 12, the CubeWerx WOS assigns an identifier when an association is created.

8.12.3. Methods

The following table defines the behavior of the /associations and /{association URL} resources with respect to the specified HTTP methods.

NOTE	As mentioned earlier, the URL template used by the CubeWerx WOS for an association is /associations/{id}.

Table 9. Methods

Resource	Method	Action
/associations	GET	Gets the specified association(s) based on the values of the specified query parameters
/associations	POST	Creates a new association; the body of the request contains a representation of the association encoded using the schema defined in this clause.
/associations	PUT	Must be specified with a predicate to define a subset of associations to be updated; the content of the request contains a representation of the replacement content
/associations	DELETE	Must be specified with a predicate to define a subset of associations to be removed
/{association URL}	GET	Gets the specified association
/{association URL}	POST	Undefined
/{association URL}	PUT	Updates an existing association; the body of the request contains a representation of the replacement association
/{association URL}	DELETE	Removes an association from the system

8.12.4. Examples

Example 1: Get all the associations for a specific relationship; renders in this example.

```
CLIENT                                                    SERVER
  |                                                         |
  |   GET /1.0/wos/associations?rel=urn:ogc:associationType:renders" HTTP 1.1
  |   Host: www.someserver.com                              |
  |-------------------------------------------------------->|
  |                                                         |
  |   HTTP/1.1 200 OK                                       |
  |   Content-Type: application/ogc-as+xml                  |
  |                                                         |
  |   <?xml version="1.0"?>                                 |
  |   <Associations> version="1.0.0">                       |
  |      <Association rel="...:renders">                    |
  |         <Source type="...::wms:layer"                   |
  |            xlink:href="https://tb12.cubewerx.com/a011/cubeserv">
  |            <Identifier>USGS.Elev_Contour</Identifier>   |
  |         </Source>                                       |
  |         <Target type="...::wfs:featureType"             |
  |            xlink:href="https://tb12.cubewerx.com/a011/cubeserv?DATASTORE=USGS">
  |            <Identifier namespace=".../">Elev_Contour</Identifier>
  |         </Target>                                       |
  |      </Association>                                     |
  |      <Association rel="...:renders">                    |
  |         <Source type=".../wms:layer"                    |
  |           xlink:href="https://tb12.cubewerx.com/a011/cubeserv">
  |            <Identifier>USGS.CellGrid_7_5Minute</Identifier>
  |         </Source>                                       |
  |         <Target type="...::wfs:featureType"             |
  |            xlink:href="https://tb12.cubewerx.com/a011/cubeserv?DATASTORE=USGS">
  |            <Identifier namespace="...">CellGrid_7_5Minute</Identifier>
  |         </Target>                                       |
  |      </Association>                                     |
  |      ...                                                |
  |   </Associations>                                       |
  |<--------------------------------------------------------|
```

Example 2: Create an association object with a description and dynamic properties.

```
CLIENT                                                    SERVER
  |                                                         |
  |   POST /1.0/wos/associations  HTTP 1.1                  |
  |   Host: www.someserver.com                              |
  |   Content-Type: application/ogc-as+xml                  |
  |                                                         |
  |   <?xml version="1.0" encoding="UTF-8"?>                |
  |   <wos:Association                                      |
  |      rel="urn:ogc:def:associationType:OGC-CSW-ebRIM::renders"
  |      xmlns:wos="http//www.opengis.net/wos/1.0"          |
  |      xmlns:xlink="http://www.w3.org/1999/xlink"         |
  |      xmlns:gml="http://www.opengis.net/gml/3.2"         |
```

```
     |      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"|
     |      xsi:schemaLocation="http://www.w3.org/1999/xlink     |
     |                http://www.w3.org/1999/xlink.xsd           |
     |                http://www.opengis.net/gml/3.2             |
     |                http://www.pvretano.com/schemas/gml/3.2.1/gml.xsd
     |                http://www.opengis.net/wos/1.0             |
     |                http://www.pvretano.com/schemas/wos/1.0.0/wos.xsd">
     |      <wos:Description key="title">The association's title.</wos:Description>
     |      <wos:Description key="abstract">Some narrative about this
association.</wos:Description>
     |      <wos:Property name="prop01" type="xsd:string">Value01</wos:Property>
     |      <wos:Property name="geom" type="gml:Polygon">        |
     |         <gml:Polygon                                      |
     |            srsName="http://www.opengis.net/def/crs/epsg/0/4326" gml:id="P1">
     |            <gml:exterior>                                 |
     |               <gml:LinearRing>                            |
     |                  <gml:posList>-19.06099128723145 -169.9416961669922
-19.05653190612793 -169.9346008300781 -19.0523681640625 -169.9278564453125
-19.04729080200195 -169.9230346679688 -19.03918266296387 -169.9215698242188
-19.04058837890625 -169.9138641357422 -19.04656600952148 -169.9136047363281
-19.05992698669434 -169.9196014404297 -19.06432342529297 -169.9275665283203
-19.06826400756836 -169.9364929199219 -19.06099128723145
-169.9416961669922</gml:posList>
     |               </gml:LinearRing>                          |
     |            </gml:exterior>                                |
     |         </gml:Polygon>                                    |
     |      </wos:Property>                                      |
     |      <wos:Source type="urn:ogc:def:objectType:OGC-CSW-ebRIM::wms:layer"
     |         xlink:href="https://tb12.cubewerx.com/a011/cubeserv">
     |         <Identifier>USGS.Elev_Contour</Identifier>        |
     |      </wos:Source>
     |      <wos:Target type="urn:ogc:def:objectType:OGC-CSW-ebRIM::wfs:featureType"
     |         xlink:href="https://tb12.cubewerx.com/a011/cubeserv?DATASTORE=USGS">
     |         <Identifier
namespace="http://schemas.cubewerx.com/namespaces/null">Elev_Contour</Identifier>
     |      </wos:Target>                                        |
     |   </wos:Association>                                      |
     |---------------------------------------------------------->|
     |                                                           |
     |   HTTP/1.1 200 OK                                         |
     |   Location: /1.0/wos/associations/12345                   |
     |<----------------------------------------------------------|
```

8.13. Classifications

8.13.1. Introduction

Objects managed by a WOS may be classified using multiple taxonomies. The resources defined in this clause allow taxonomies or classification schemes to be created and managed by the WOS and

also allow objects to be tagged with nodes from those classifications schemes thus supporting taxonomic queries.

NOTE	No support for taxonomies was implemented in CubeWerx's TB12 WOS because the proposed demo scenario was primarily focus on general objects and associations.

8.13.2. Resources

This engineering report defines the following resources to support the management and use of taxonomies:

- /classificationSchemes
  - /{classification scheme URL}
- /classificationNodes
  - /{classification node URL}
- /classifications
  - /{classification URL}

NOTE	The classification scheme, classification node and classification URLs are opaque but for testbed 12, CubeWerx's WOS used the following URL templates for these resources: /classificationSchemes/{id}, /classificationNodex/{id} and /classifications/{id}.

Query parameters

The following table summarizes the query parameters that may be used with the //classificationSchemes, /classificationNodes and /classifications resources.

Table 10. Object query parameters

Parameter	Description
outputFormat	Specifies the output representation for the corresponding resource
id	a comma-separated list of identifiers

NOTE	The outputFormat query parameter may also be employed with the /{classification scheme URL}, /{classification node URL} and /{classification URL} resources.

Representations

Classification Schemes

The following XML schema fragment defines the XML-encoding for a classification schemes or taxonomies:

```
<xsd:element name="ClassificationSchemes">
   <xsd:complexType>
      <xsd:complexContent>
         <xsd:sequence>
            <xsd:element ref="wos:ClassificationScheme"
                         minOccurs="0" maxOccurs="unbounded"/>
         </xsd:sequence>
         <xsd:attribute name="version" type="xsd:string"
                        use="required" fixed="1.0.0"/>
      </xsd:complexContent>
   </xsd:complexType>
</xsd:element>

<xsd:element name="ClassificationScheme">
   <xsd:complexType>
      <xsd:complexContent>
         <xsd:extension base="assoc:BaseElementType">
            <xsd:sequence>
               <xsd:element ref="wos:ClassificationNode"
                            minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="id" type="xsd:anyURI" use="required"/>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>
</xsd:element>
```

The ClassificationSchemes element is a container for all the classification schemes offered by a WOS.

A classification scheme is the root node of a hierarchically related set of classification nodes. The *id* parameter is the unique identifier for the classification scheme.

The following XML schema fragment defines the XML-encoding for a classification node.

```
<xsd:element name="ClassificationNodes">
   <xsd:complexType>
      <xsd:complexContent>
         <xsd:sequence>
            <xsd:element ref="wos:ClassificationScheme"
                         minOccurs="0" maxOccurs="unbounded"/>
         </xsd:sequence>
         <xsd:attribute name="version" type="xsd:string"
                        use="required" fixed="1.0.0"/>
      </xsd:complexContent>
   </xsd:complexType>
</xsd:element>

<xsd:element name="ClassificationNode">
   <xsd:complexType>
      <xsd:complexContent>
         <xsd:extension base="assoc:BaseElementType">
            <xsd:sequence>
               <xsd:element name="code" type="xsd:string" minOccurs="0"/>
               <xsd:element name="path" type="xsd:string" minOccurs="0"/>
            </xsd:sequence>
            <xsd:attribute name="id" type="xsd:anyURI" use="required"/>
            <xsd:attribute name="parent" type="xsd:anyURI" use="required"/>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>
</xsd:element>
```

The ClassificationNodes element is an unordered container for all the classification nodes offered by a WOS.

Each classification node's unique identifier is encoded using the *id* parameter. The identifier of the node's direct parent node in a hierarchy is encoded using the *parent* parameter. The parent identifier of the level 1 nodes in a taxonomy is the identifier of the classification scheme.

A unique code string may be assigned to each node using the *code* parameter. A string representation of the node's path may be included in the definition of the node using the *path* element.

Classifications

The following XML schema fragment defines the XML-encoding for classifications:

```
<xsd:element name="Classifications">
   <xsd:complexType>
      <xsd:complexContent>
         <xsd:sequence>
            <xsd:element ref="wos:Classification"
                         minOccurs="0" maxOccurs="unbounded"/>
         </xsd:sequence>
         <xsd:attribute name="version" type="xsd:string"
                        use="required" fixed="1.0.0"/>
      </xsd:complexContent>
   </xsd:complexType>
</xsd:element>

<xsd:element name="Classification">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element name="ClassifiedObjectId" type="xsd:anyURI"/>
         <xsd:element name="ClassificationScheme"
                      type="xsd:anyURI" minOccurs="0"/>
         <xsd:element name="ClassificationNode" type="xsd:anyURI"/>
      </xsd:sequence>
      <xsd:attribute name="id" type="xsd:anyURI" use="required"/>
   </xsd:complexType>
</xsd:element>
```

The *Classifications* element is a container for all the classifications maintained by a WOS.

A classification is a specialized association between an object managed by a WOS and a node in a classification scheme. Each classification has a unique identifier encoded using the *id* parameter.

The *ClassifiedObjectId* parameter encodes the identifier of the WOS object being classified.

The *ClassifiedNode* parameter encodes the identifier of the classification nodes being used to classify the specified object.

The optional *ClassificationScheme* parameter is used to encode the identified of the classification scheme used to classify the specified object. It is included for lookup efficiency since this information can be derived by traversing the classification hierarchy.

8.13.3. Methods

The following table defines the behavior of the /classificationSchemes, /classificationNodes++ and /classifications resources with respect to the specified HTTP methods.

Table 11. Methods

Resource	Method	Action
/classificationSchemes	GET	With an *id* query parameter, retrieves classification schemes with the specified identifiers; otherwise retrieves all the classification schemes offered by a WOS
/classificationSchemes	POST	Creates a new classification scheme; the body of the request shall contain a representation of a *classificationScheme*
/classificationSchemes	PUT	An *id* query parameter must be specified enumerating a subset of classification schemes to be updated; the body of the request shall contain a representation of the replacement *classificationScheme*
/classificationSchemes	DELETE	Ad *id* query parameter must be specified enumeration a subset of classification scheme to be deleted; all non-shared classification nodes and classifications using this taxonomy shall be deleted as well
/classificationNodes	GET	With an *id* query parameter, retrieves classification nodes with the specified identifiers; otherwise retrieves all the classification node offered by a WOS
/classificationNodes	POST	Creates a new classification node; the body of the request shall contain a representation of a *classificationNode*
/classificationNodes	PUT	An *id* query parameter must be specified enumerating a subset of classification nodes to be updated; the body of the request shall contain a representation of the replacement *classificationNode*
/classificationNodes	DELETE	An *id* query parameter must be specified enumerating a subset of classification nodes to be deleted
/classification	GET	With an *id* query parameter, retrieves classifications with the specified identifiers; otherwise retrieves all the classifications managed by a WOS
/classification	POST	Undefined
/classification	PUT	An *id* query parameter must be specified enumerating a subset of classifications to be updated; the body of the request shall contain a representation of the replacement *classification*

Resource	Method	Action
/classification	DELETE	An *id* query parameter must be specified enumerating a subset of classifications to be deleted
/{classification scheme URL}	GET	Retrieves the specified classification scheme
/{classification scheme URL}	POST	Undefined
/{classification scheme URL}	PUT	Updates the specified classification scheme; the body of the request shall contain a representation of the replacement *classification scheme*
/{classification scheme URL}	DELETE	Deletes the specified classification scheme, all unshared classification nodes and any classifications using this taxonomy
/{classification node URL}	GET	Retrieves the specified classification node
/{classification node URL}	POST	Undefined
/{classification node URL}	PUT	Updates the specified classification node; the body of the request shall contain a representation of the replacement *classification node*
/{classification node URL}	DELETE	Deletes the specified classification node
/{classification}	GET	Retrieves the specified classification
/{classification}	POST	Undefined
/{classification}	PUT	Updates the specified classification; the body of the request shall contain a representation of the replacement *classification*
/{classification}	DELETE	Deletes the specified classification node

8.13.4. Examples

Example 1: Get all the classification schemes offered by a WOS.

```
CLIENT                                                    SERVER
  |                                                         |
  |   GET /1.0/wos/classifications                          |
  |   Host: www.someserver.com                              |
  |-------------------------------------------------------->|
  |                                                         |
  |   HTTP/1.1 200 OK                                       |
  |   Content-Type: application/ogc-cl+xml                  |
  |                                                         |
  |   <?xml version="1.0" encoding="UTF-8"?>                |
  |   <ClassificationSchemes                                |
  |      xmlns="http://www.opengis.net/wos/1.0"             |
  |      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"|
  |      xsi:schemaLocation="http://www.opengis.net/wos/1.0  |
  |                         ../wos.xsd">                    |
  |      <ClassificationScheme                              |
  |         id="urn:ClassificationScheme:ISO-19119:Services"> |
  |         <ClassificationNode                             |
  |            parent="urn:ISO-19119:Service:Info-Management" |
  |            id="urn:ISO-19119:Service:Feature-Type">     |
  |            <code>Feature-Type</code>                    |
  |         </ClassificationNode>                           |
  |         <ClassificationNode                             |
  |            parent="urn:ISO-19119:Service:Info-Management" |
  |            id="urn:ISO-19119:Service:Catalogue">        |
  |            <code>Catalogue</code>                       |
  |         </ClassificationNode>                           |
  |         <ClassificationNode                             |
  |            parent="urn:ISO-19119:Service:Info-Management" |
  |            id="urn:ISO-19119:Service:Registry">         |
  |            <code>Registry</code>                        |
  |         </ClassificationNode>                           |
  |         <ClassificationNode                             |
  |            parent="urn:ISO-19119:Service:Info-Management" |
  |            id="urn:ISO-19119:Service:Gazetteer">        |
  |            <code>Gazetteer</code>                       |
  |         </ClassificationNode>                           |
  |         .                                               |
  |         .                                               |
  |         .                                               |
  |      <ClassificationScheme/>                            |
  |      .                                                  |
  |      .                                                  |
  |      .                                                  |
  |   <ClassificationSchemes>                               |
  |<--------------------------------------------------------|
```

Example 2: Classify an object as a feature type

```
   CLIENT                                                    SERVER
     |                                                         |
     |   POST /1.0/wos/classification  HTTP 1.1                |
     |   Host: www.someserver.com                              |
     |   Content-Type: application/ogc-cl+xml                  |
     |                                                         |
     |   <?xml version="1.0" encoding="UTF-8"?>                |
     |   <Classification                                       |
     |      xmlns="http://www.opengis.net/wos/1.0"             |
     |      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"|
     |      xsi:schemaLocation="http://www.opengis.net/wos/1.0 ../wos.xsd">
     |   <ClassifiedObjectId>urn:uuid:ce28abbe-d17e-426d-8dae-
a7977442a7be</ClassifiedObjectId>
     |   <ClassificationNode>urn:ISO-19119:Service:Feature-Type</ClassificationNode>
     |   </Classification>                                     |
     |                                                         |
     |-------------------------------------------------------->|
     |                                                         |
     |   HTTP/1.1 200 OK                                       |
     |   Location: /1.0/wos/classifications/12345              |
     |<--------------------------------------------------------|
```

Chapter 9. Future Work

This section summaries the key findings and defines work items that are recommended for future OGC Innovation Program initiatives to further analyze the GFM capabilities.

The GFM activity has been a proof of concept, targeting the initial requirements from two different implementation angles. Both approaches have successfully demonstrated that it is very well possible to accommodate non-geospatial information and it's associations with related data as features in the GFM.

So what is next? This testbed experimented with the GFM as such, did an in-depth analysis of the UML model and its implementation, and verified that the GFM can serve well as an association model between any type of features. That is, for example, a video stream can be associated to a Twitter message, or an individual feature in a database can be associated with the results of a complex stream processing workflow. The associations can be features themselves that, if necessary, associate further association features. This approach allows for graphs of features and association features, which is an extremely powerful approach that needs to be further explored. In order to further demonstrate that the concept is a viable approach and the capabilities of the GFM go well beyond the traditional geoinformation-centric application areas it was originally designed for, a larger scale interoperability experiment would be required. The following aspects require further research:

- **Analogy to Linked Data**: The feature to feature with association is pretty similar to the Linked Data approach that uses triplets to describe associations between features. Just, the GFM approach is even more powerful because the predicate is not constrained but a first class object in its own right. How does this extend the capabilities of Linked Data and how does it allow proper integration with existing Linked Data implementations?
- **Semantic of Association Features**: The approach documented in this engineering report becomes more powerful if the semantics of the association features are well defined. The Semantic Web domain uses ontologies to define the semantics behind objects. Semantic Web approaches should be analyzed to better understand how the full potential of the GFM approach can be leveraged.
- **Integration of different semantics**: The flexibility of the approach allows for different semantics to be incorporated into knowledge graphs (that consist of features and associations features). Future activities need to analyze how these different semantics can be catered for. Providing access to ontologies is certainly a first step, but potential access and availability issues need to be handled.
- **Definition of a core set of association features**: The development of a core set of association types and corresponding association features would help the geospatial domain to make use of the model more efficiently. A process needs to be developed that allows the OGC community to develop, maintain, and provide this core set. The OGC Naming Authority needs to be part of that process.
- **Implementation details**: The Testbed-12 research has used a limited subset of possible data types and access mechanisms. It is recommended that future initiatives include aspects such as:
    - A greater variety of data types and formats

- Software analytics generating new information
- Real time data feeds
- More complex exercise scenarios
- Interoperability experiments across the implementation approaches
- Defined client requirements and resources

Appendix A: Revision History

Table 12. Revision History

Date	Release	Editor	Primary clauses modified	Descriptions
June 30, 2016	0.1	M. Klopfer	all	draft outline
Sept 30, 2016	0.2	M. Klopfer	all	draft engineering report
Oct 24, 2016	0.3	M. Klopfer	all	final draft
Oct 31, 2016	0.4	M. Klopfer	all	milestone draft
Nov 15, 2016	0.5	M. Klopfer	all	final comments

Appendix B: Bibliography

[1] Roswell, Charles: Modeling of Geographic Information. In: Kresse, Wolfgang and D. Danko: Springer Handbook of Geographic Information. Springer, 2012